RIGA TECHNICAL UNIVERSITY

Faculty of Electrical and Enviromental Engineering

Pablo Beléndez Pascual

DEVELOPMENT OF A DEMONSTRATION TEST BENCH BASED ON A PLC AND HMI

BACHELOR'S THESIS

Supervisor

Asoc. Prof. Ingars Steiks

RIGA 2023

TABLE OF CONTENTS

LIST OF SYMBOLS

AC – Alternating Current

CPU – Central Processing Unit

DC – Direct Current

GUI – Graphical User Interface

HMI – Human-Machine Interface

I/O – Input/Output

IP – Internet Protocol Address

LAN – Local Area Network

LED – Light-Emitting Diode

mA – Milliampere

No – Number

PLC – Programmable Logic Controller

RAM – Random Access Memory

ROM - Read-Only Memory

V – Volts

VDC – Volts Direct Current

ABSTRACT

This paper examines the development of a test bench based on PLC, HMI, and external components. This setup will be tested with several software programs in order to achieve a full interaction with all the components. The present study provides deep learning about the hardware by analyzing its characteristics and scope in more detail. Also, it provides a profound knowledge of the software functions, commands, and initialization. Furthermore, this thesis contains many introductory examples involving digital and analog I/O, such as turning on a LED and manually controlling the input voltage range. After all, a parking garage simulation, which involves all the hardware, will be built as a real PLC and HMI application.

To sum up, this work can be a manual for future students to show how to program this test bench and get the required knowledge to develop real-world applications.

Key words: PLC, HMI, software, sensors, configuration, input, output, voltage.

ANOTĀCIJA [LV]

Šajā rakstā ir apskatīta testa stenda izstrāde, pamatojoties uz PLC, HMI un ārējiem komponentiem. Šī iestatīšana tiks pārbaudīta ar vairākām programmatūras programmām, lai panāktu pilnīgu mijiedarbību ar visiem komponentiem. Šis pētījums sniedz dziļas zināšanas par aparatūru, detalizētāk analizējot tās īpašības un apjomu. Tas arī nodrošina padziļinātas zināšanas par programmatūras funkcijām, komandām un inicializāciju. Turklāt šajā darbā ir iekļauti daudzi ievada piemēri, kas saistīti ar digitālo un analogo I/O, piemēram, gaismas diodes ieslēgšana un manuāla ieejas sprieguma diapazona kontrole. Galu galā autostāvvietas simulācija, kas ietver visu aparatūru, tiks izveidota kā īsta PLC un HMI lietojumprogramma,

Rezumējot, šis darbs var būt rokasgrāmata topošajiem studentiem, lai parādītu, kā programmēt šo pārbaudes stendu un iegūt nepieciešamās zināšanas, lai izstrādātu reālās pasaules lietojumprogrammas.

Atslēgas vārdi: PLC, HMI, programmatūra, sensori, konfigurācija, ieeja, izeja, spriegums.

INTRODUCTION

This Project is based on setting up a test bench including a PLC, an HMI and external components. Recently, PLCs and HMIs are taking a step forward in industrial automation and control systems, so the idea is to get more information about them. In this thesis some introductory sections related to PLC and HMI are argued, with the aim of knowing their definition, history, operation, components, configuration... On the other hand, external components have been added to the configuration by attaching them to the PLC or through a circuit board directly connected to the PLC. Thus, these components are able to interact with the PLC and the HMI thanks to the development of specific software, allowing the user to simulate real-world applications. It can be shown in the project from basic examples to an extensive simulation of a car park that can be emulated with the computer, PLC, HMI, or with all of them together. This work is also a manual for future students who want to improve their knowledge on this subject. Therefore, it includes the steps to initialise both CX-Programmer and NB-Designer software, as well as the main commands in the program are explained with a variety of examples.

Basically, the thesis not only presents a complete project, but also serves as a practical guide for students interested in expanding their knowledge of PLCs, HMIs and related software.

1. HARDWARE

1.1. PLC

A PLC is a ruggedized industrial computer designed to perform and control different automated processes, especially for industrial applications. As a result, PLCs have become a fundamental tool in order to develop industrial technological processes. In my case, I will test the “CJ2M” OMRON PLC as seen in the Figure 1.1.

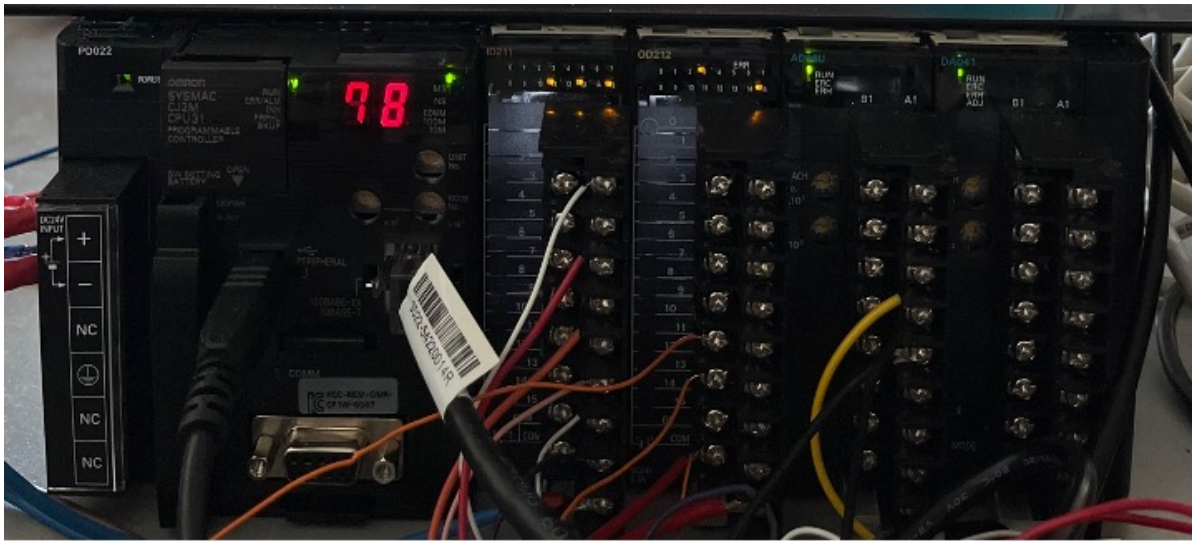

Fig. 1.1. OMRON PLC

1.1.1. Components and functioning

Broadly speaking, a PLC is composed of few basic parts. Despite the fact that they might look slightly different from various manufacturers, the purpose and scope of each component are the same.

Firstly, its power supply has to be mentioned, which converts the AC taken from the outlet to unregulated DC suitable for the PLC and its components. Secondly, the CPU appears, which is responsible for executing the user-defined program. It processes input signals, performs logic and arithmetic operations, and sends output signals to the connected devices. Moreover, related to PLC racks, it is a hardware assembly that house and organizes the various components of a PLC system. In addition, the I/O Modules are used to connect the processor to the field devices. So, an input module detects the status of input signals such as switches, push-buttons, temperature sensors… On the other hand, an output module

controls devices such as lights, relays, motor starters, etc. Commonly, depending on the specific application, I/O modules may be digital or analog. In Figure 1.2 can be shown a typical PLC I/O system connection.

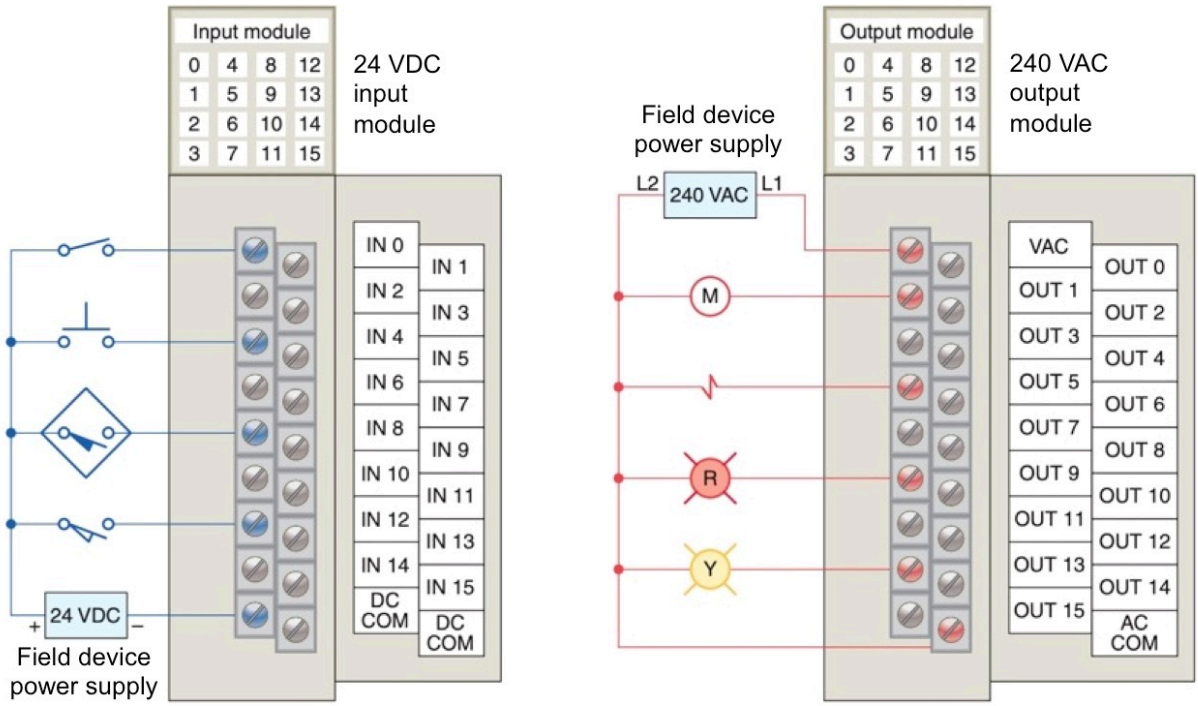

Fig. 1.2. Typical PLC I/O system connection, (a) Analog input module and (b) analog output module

Furthermore, the Communication Interface allows the PLC to communicate with other devices, such as a computer or an HMI. It may support various communication protocols, namely Modbus, Profibus, Ethernet/IP, and others [1]. The common types of memory used in PLCs are ROM, where programs and data are stored, and RAM, where data can uniquely be read and not written [2]. Finally, PLC is programmed by a program or software configured by a computer, as a result, it will be loaded into the PLC. It can be shown in Figure 1.3 a PLC typical function and operation.

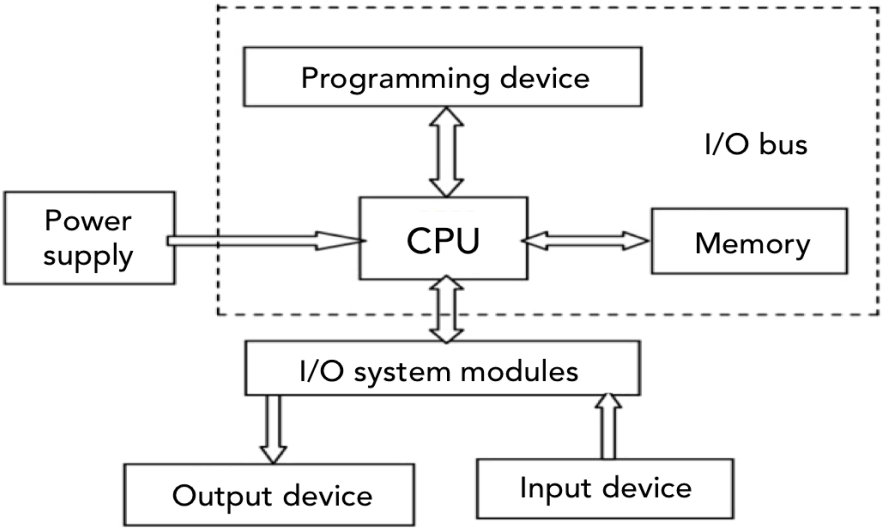

Fig. 1.3. Typical function and operation of a PLC

1.1.2. Racks

The rack serves as the backbone of the modular PLC system that holds all the modules together such as the CPU, Power Supply, Communication Module, I/O Modules, and additional function modules, in a single frame. PLC racks work by receiving input signals from sensors and switches, processing them using a program logic that is stored in the CPU, and then sending output signals to control devices in the system (Figure 1.4).

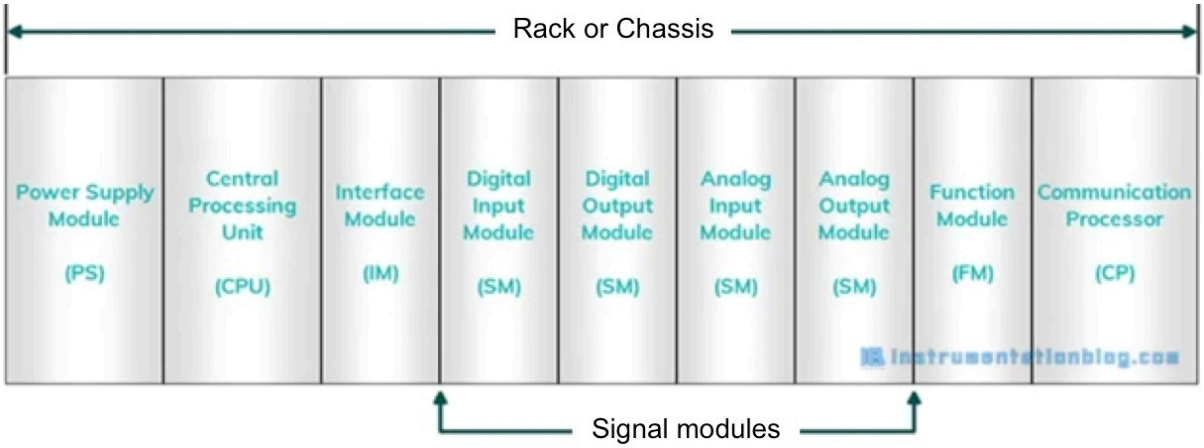

Fig. 1.4. Different modules that are fitted in the same rack or chassis of the modular PLC system. Image source: instrumentationblog.com

The four different racks within the PLC are the Digital input, Digital output, Analog input, and Analog output. Each has its own characteristics and purposes. Generally, an input module detects the status of input signals such as push-buttons, switches, temperature

sensors, etc. An output module controls devices such as relays, motor starters, LEDs, etc. Differentiating digital and analog modules, digital I/O are referred to signals that are either ON or OFF, 1 or 0, true or false, open or closed, so simply signals to process for a PLC. On the contrary, Analog I/O specifies signals that have a range of values much greater than just 1 or 0. For instance, an analog signal could produce a voltage anywhere in the range of 0 – 10 VDC. The signal could be 2 V, 3 V, 8.5 V, etc.

Each digital rack of our PLC has a set of 16 channels, where external components can be attached, and a set of 16 LED indicators visually indicating the status of each bit. It is the same for analog racks, they have a set of 16 channels, to which external components can be attached, however without LED indicators. On the contrary, depending on in which exactly bits are these components attached, will be different signal ranges such as 1-5V, 0-10V, or −10-10V. The first rack starting from the left is the digital input, the second is the digital output, and then the analog input and analog output [3].

1.1.3. History

The first PLC was developed by Dick Morley and Bedford Associates in the 1960s, see Figure 1.5. PLC was developed as a solution to replace large banks of hardwired relays and timers, performing more operations in a fraction of the time. There were several advantages of using digital programmable controls as their small size and enormous power efficiency. In addition, the expected lifetime of a system would certainly be much longer, as solid-state devices, e.g. transistors, last longer. The time required to make repairs or changes to the system would absolutely decrease, due to the changes being directly downloaded from the computer to the controller.

In the beginning, the original specifications that were required for the PLC included a few inputs and outputs, and a small memory. PLCs were simply known by the name Modicon 084 or “Programmable Controller”. Then, the Allen-Bradley company popularized the PLC name and acronym in the early 1970s. During the following years, more companies entered the PLC market and the programming software also advanced. From simple keypads allowing direct text entry, to modern software on operating systems that can allow the user to program and carefully document an entire process easily [4].

Fig. 1.5. Historical photo showing from left to right: Dick Morley, Tom Bois-sevain, Modicon 084, George Schwenk, and Jonas Landau. Image source: Automation.com

1.1.4. Applications

In today's world, PLC systems can be found everywhere, including factories, office buildings and even controlling the traffic on the streets. PLCs are the very heart of the control of many critical technologies that most people do not even imagine, as a result, they are so seamlessly and invisibly integrated into people's life. Here are some examples of everyday mechanical systems that are controlled by PLCs [5], see Figures 1.6 and 1.7.

- Road Traffic Signals
- The Automatic Car Wash
- The elevator
- Automatic Doors
- Parking garage
- Conveyor belts

Fig. 1.6. Road traffic signal. Credits: FREEPIK

Fig. 1.7. Conveyor belts. Image source: algevasa.com

1.2. HMI

HMI refers to a dashboard that enables a user to interact with a controller. Basically, HMI allows users to visualize data about operations and control machinery. Similar to how you would interact with your air-conditioning system to check and control the temperature in your house, a plant-floor operator might use an HMI to check and control the temperature of an industrial water tank, or to see if a certain pump in the facility is currently running [6].

Generally speaking, depending on the type of manufacturer, HMI might be composed of a screen where it shows some visual information about the software in progress. In my case I will run the "NB10W-TW01B" OMRON HMI model as seen in Figure 1.8.

Fig. 1.8. OMRON HMI

1.3. TCP/IP connections

This test bench groups together all the involved hardware via Ethernet. Basically, Ethernet is a widely used technology for local area networks that allows devices to communicate with each other over a wired connection. The connection between the PLC, HMI, and computer via Ethernet typically involves establishing a communication network that allows these devices to exchange data and commands.

Firstly, we should connect the PLC to the Ethernet network by connecting an Ethernet cable from the PLC's Ethernet port to an available port on the LAN. In addition, we should configure the PLC's Ethernet settings by assigning a unique IP address, 192.168.1.120. The next step is to connect the HMI to the Ethernet network. This is done by connecting an Ethernet cable from the HMI's Ethernet port to an available port on the LAN. Also, we must configure the Ethernet settings on the Omron HMI to match the network configuration. This includes assigning a unique IP address, 192.168.1.120, and a subnet mask, 255.255.255.0. The computer should also be connected to the same Ethernet network as the Omron PLC and HMI. This is done by connecting an Ethernet cable from the computer's Ethernet port to an available port on the LAN. Another possibility might be to connect the computer via WIFI with the Ethernet network, but in my case, it will be done via Ethernet cable.

Once these steps have been completed, the computer, Omron PLC, and Omron HMI should be able to communicate with each other via Ethernet. The user can then use the software on the computer to program and control the PLC and HMI [7], Figure 1.9.

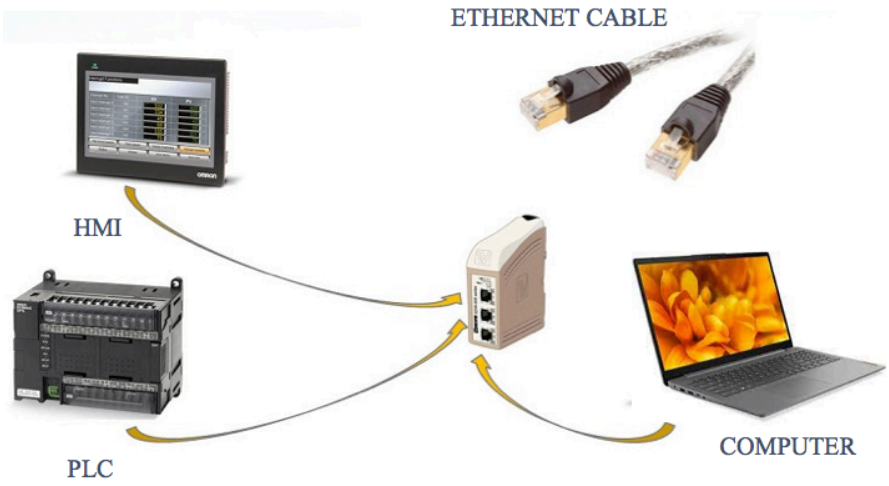

Fig. 1.9. ETHERNET connection

1.4. External components

In this project, some external components have been attached to the PLC with the aim of testing several functions. Some of these components have been attached directly to the PLC or through a circuit board.

- Switch: An electrical switch is a device used to establish or interrupt the flow of electrical current in a circuit. It is commonly used to control the ON/OFF state of a circuit or to redirect the current to different paths. The type of switch argued in this thesis is the "Push-button" Switch. This type of switch returns to its original position when released. It is frequently used for momentary operations, such as turning on a light or activating a device only while the button is pressed [8].
- Resistor: A resistor restricts or impedes the flow of electric current in a circuit. It has a specific resistance value that determines how much the resistor resists the flow of current, measured in ohms. Speaking about the analog resistor, it is used to control the voltage and current levels in a circuit. [9].
- Motor: A motor is a device that converts electrical energy into mechanical energy in order to create a rotational movement [10].
- Transistor switch: A transistor switch controls the flow of current through the load by switching itself on or off.
- Diode: In this thesis, the diode protects the circuit from damaging voltage spikes. Diodes are placed in series with each of the transistors to block the reverse current and ensure that it does not flow through the transistors or cause any damage [11].
- Optocoupler: In this project, an optocoupler consists of a LED and a photodetector housed within a single package. The input side of the optocoupler is connected to the microcontroller, while the output side is connected to the high-power section of the H-bridge circuit. The optocoupler provides electrical isolation between these two sections, preventing noise, voltage spikes, or ground potential differences from affecting the control circuitry [12].

2. SOFTWARE

2.1. PLC

This thesis argues the creation of PLC software by Ladder logic. A ladder diagram is the symbolic representation of the control logic used for programming a PLC.

2.1.1. Ladder diagram

A ladder diagram, also known as ladder logic, is a graphical programming language widely used in PLCs for creating control logic, Figure 2.1. Ladder Diagram is the official name given in the international PLC programming standard IEC-61131. Ladder diagrams are composed of rungs, horizontal lines of control logic, and rails, vertical lines at the start and end of each rung. The symbols used in ladder diagrams represent different electrical and logical components such as contacts, coils, timers, counters, and other control elements. Logic expressions are useful to formulate the desired control operations, commonly used in combination with the inputs and outputs.

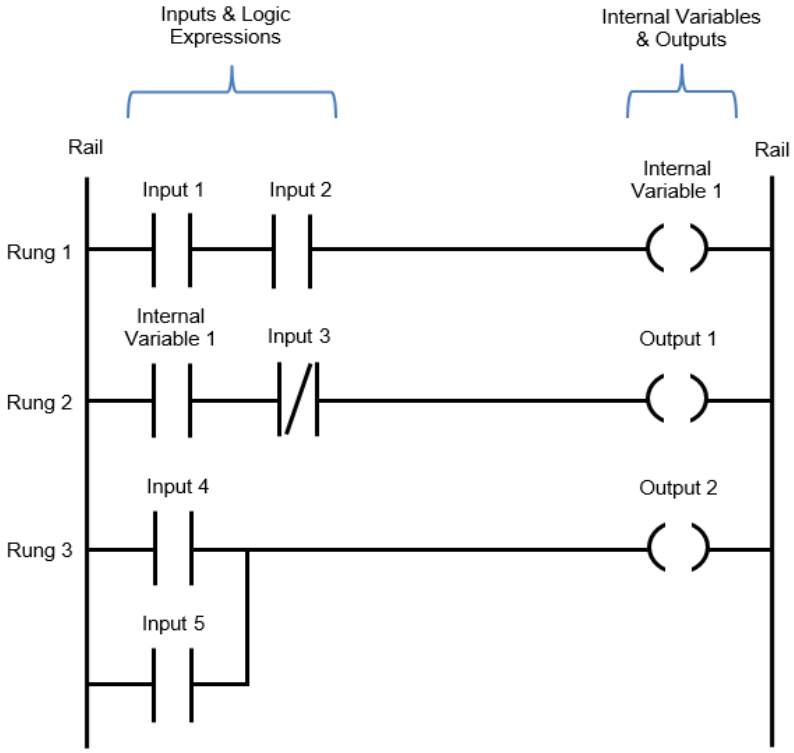

Fig. 2.1. Example of ladder diagram software. Image source: automationreadypanels.com

In addition, it is helpful to save each input, output, and expression with an address notation. This means that these addresses will always refer to each item. The comments are an extremely important part of a ladder diagram. Comments are displayed at the start of each rung and are used to describe the logical expressions and control operations being executed in that rung, or groups of rungs. Understanding ladder diagrams is made a lot easier by using comments.

Regarding how ladder logic works, it operates in a similar way to relay logic, however without all the laborious relay control wiring. Basically, input and output devices are hard-wired to the PLC. On the ladder logic program, the outputs are activated depending on the status of the input signals With reference to reading ladder logic, the program is read from the left-hand rail to the right-hand rail and from the first rung to the last rung. Briefly, left to right and top to bottom. Hence, the rungs contain input symbols that either pass or block the logic flow. The result of the rung is expressed in the last symbol, known as the output.

On the other hand, mentioning the binary concept is essential within this topic. Normally, PLCs operate on the binary concept, for instance, True or False, 1 or 0, ON or OFF, High or Low, Yes or No… In a PLC, binary events are expressed symbolically using ladder logic in the form of a normally open contact (NO) and normally closed contact (NC). The event associated with a normally open contact (NO) can be TRUE or FALSE. When the event is TRUE then it is highlighted green and the logic flow can move past it to the next logic expression, such as the current flow in an electric circuit when a switch is turned on. On the contrary, the closed contact (NC) functions exactly in the same way, nevertheless, it is basically the opposite state of an event that occurs [13].

2.1.1.1. Logic functions

Ladder logic has several logic functions. The six basic, yet essential, logic functions are NOT, AND, OR, NAND, NOR, and XOR. So, depending on how we set up the diagram, we will achieve a huge variety of software programs with different highlighted inputs and outputs. Furthermore, we will be fully able to program the majority of automation control requirements.

NOT GATE (Figure 2.2, Table 2.1)

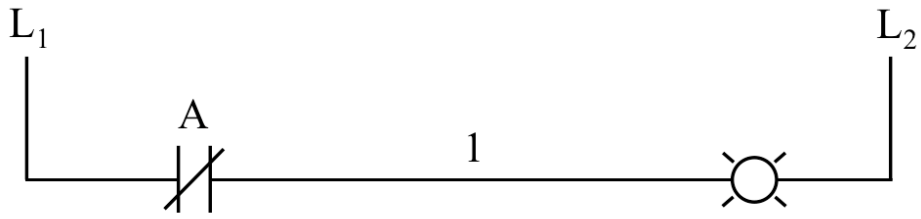

Fig. 2.2. Example of NOT function

Table 2.1.

NOT binary input and output

Input A	Output
0	1
1	0

AND GATE (Figure 2.3, Table 2.2)

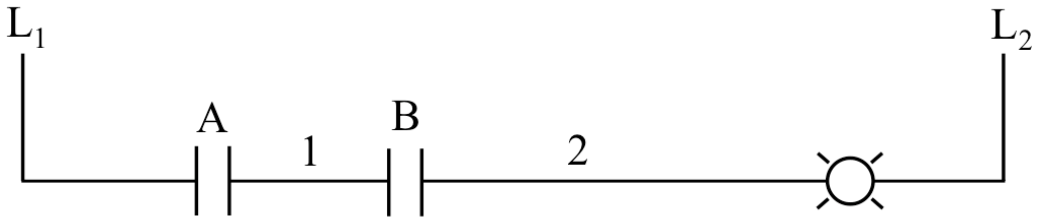

Fig. 2.3. Example of AND function

Table 2.2.

AND binary input and output

Input A	Input B	Output
0	0	0
1	0	0
0	1	0
1	1	1

The output will be activated only if both switches are activated simultaneously.

OR GATE (Figure 2.4, Table 2.3)

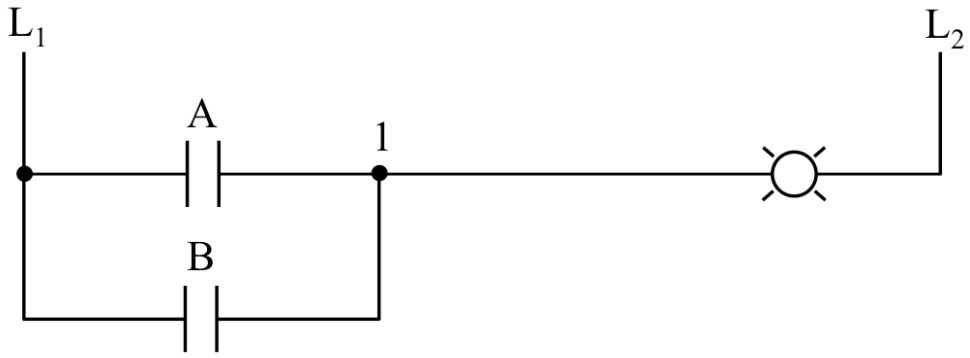

Fig. 2.4. Example of OR function

Table 2.3.

OR binary input and output

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	1

The output will be activated if any of the switches is energised.

NAND GATE (Figure 2.5, Table 2.4)

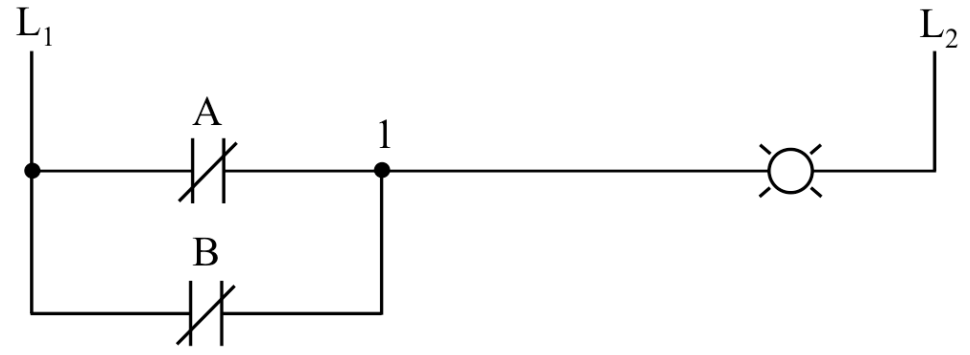

Fig. 2.5. Example of NAND function

Table 2.4.

NAND binary input and output

Input A	Input B	Output
0	0	1
1	0	1
0	1	1
1	1	0

Basically, it is the inversion of the AND gate.

NOR GATE (Figure 2.6, Table 2.5)

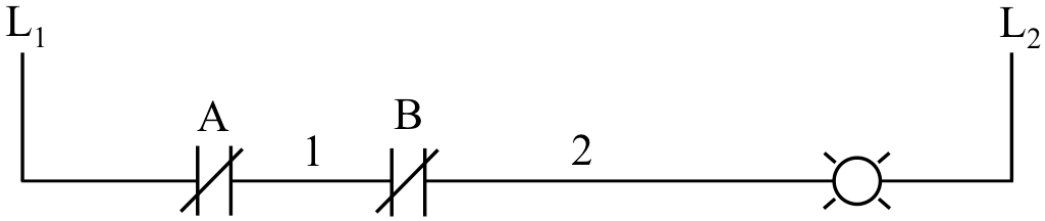

Fig. 2.6. Example of NOR function

Table 2.5.

NOR binary input and output

Input A	Input B	Output
0	0	1
1	0	0
0	1	0
1	1	0

The output will be true only if both inputs are false, the opposite of the OR gate.

XOR GATE (Figure 2.7, Table 2.6)

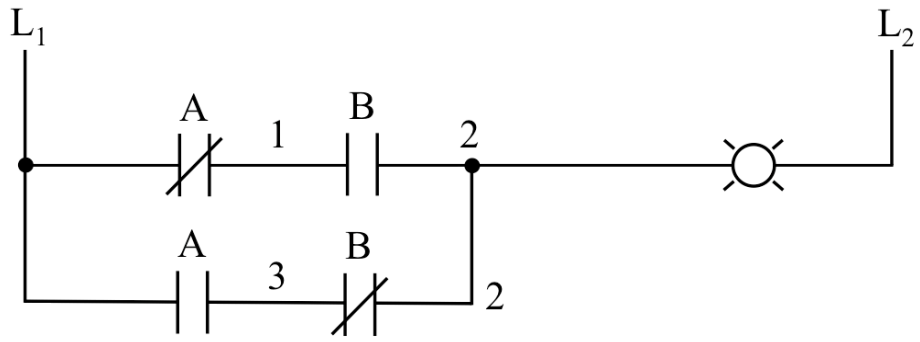

Fig. 2.7. Example of XOR function

Table 2.6.

XOR binary input and output

Input A	Input B	Output
0	0	0
1	0	1
0	1	1
1	1	0

The output will be true only if one of the inputs is true.

2.1.2. CX-Programmer initialization

In my case, I will run the project in CX-Programmer, which is wholly integrated into the CX-One software suite. It is the programming software for all of Omron´s PLC series. Before starting with the program, it is required to configure the PLC. This can be done within the pop-up window "NewPLC1[CJ2M] Offline" (Figure 2.8) [14].

Then, within "Network Settings", it is necessary to set the IP address in order to connect the computer to the PLC via Ethernet (Figure 2.9).

After that, it is required to configure the main racks within "IO Table and Unit Setup", as can be seen in Figure 2.10.

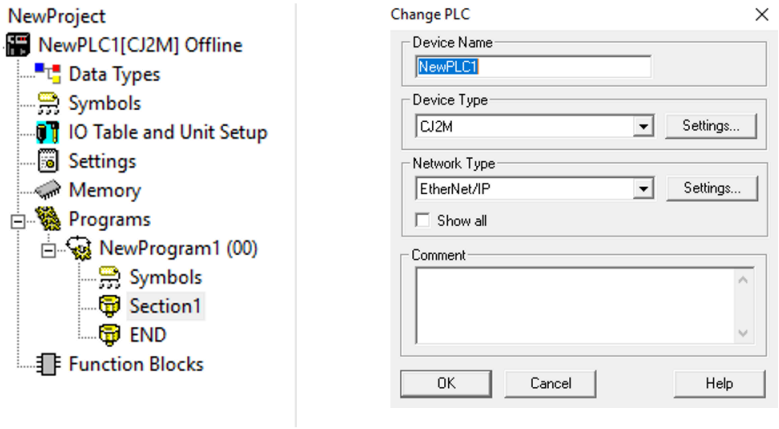

Fig. 2.8. “NewPLC1[CJ2M] Offline” setting

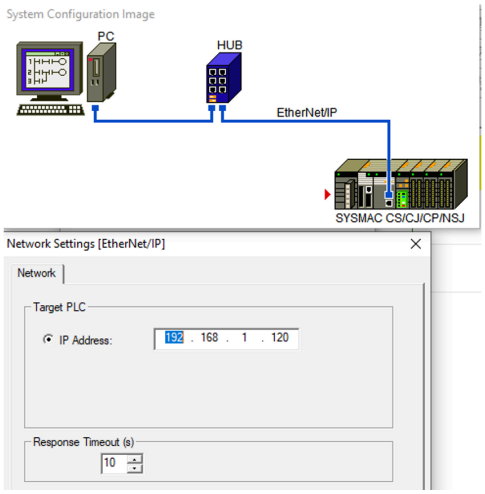

Fig. 2.9. IP address setting

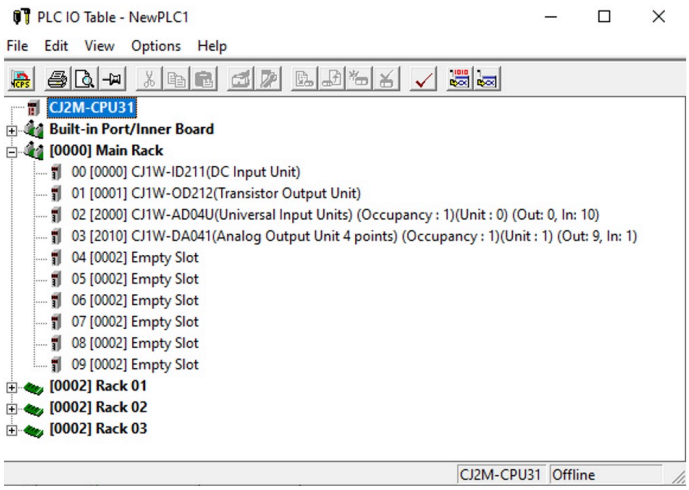

Fig. 2.10. Racks configuration

Later, while building the program, it can be tested either in simulation or in the PLC. It is required to compile the program (Figure 2.11) and afterward, if it has been successful, run it online (transfer to PLC) or through simulation.

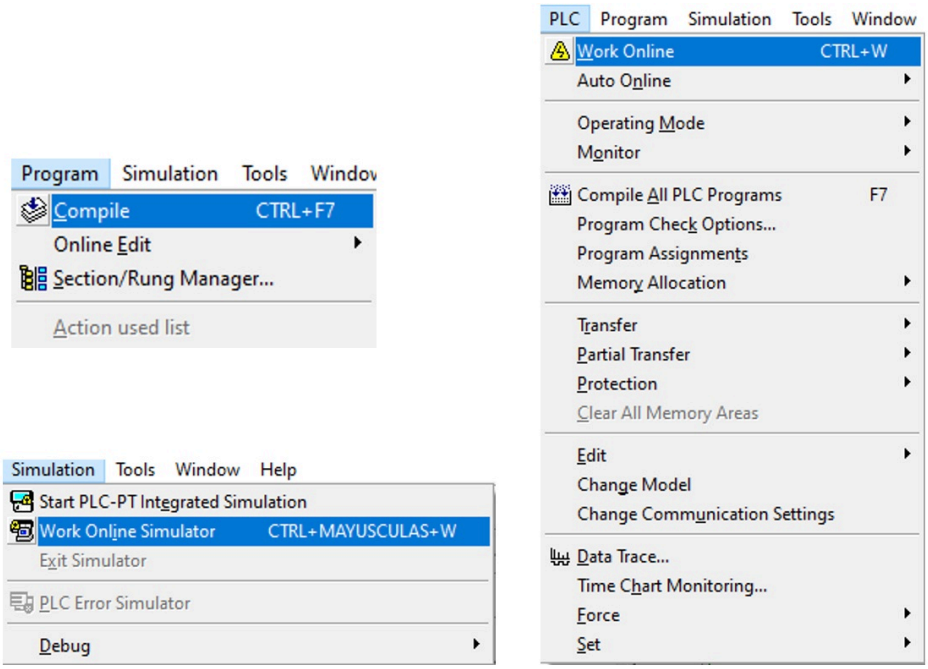

Fig. 2.11. Compiling the program

2.1.3. CX-Programmer commands

CX-Programmer provides a set of commands and instructions in order to create ladder logic programs. These are the most commonly used commands:

- **Inputs**

Inputs can appear as opened or closed contact. When it is open, the contact will be turned on when the input is activated. In contrast, closed contacts will be turned ON when the input is deactivated, shown in Figure 2.12. They are defined from 0.00 to 0.99, in defiance of our PLC will be able to show only from 0.00 to 0.15 [15].

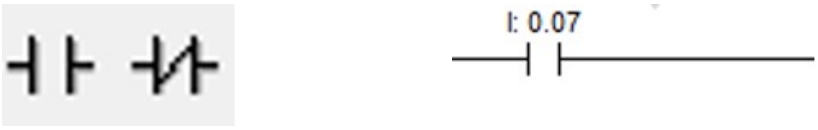

Fig. 2.12. Input command

- **Outputs**

Outputs might also appear as opened or closed coils, however, they will commonly exist as opened (Figure 2.13). They will be either energised or not depending on the input status. They are defined as 1.00, 2.00, 100.07, 200.12, and whatnot. However, our PLC will only be able to show from 1.00 to 1.15.

Fig. 2.13. Output command

- **Timers**

Timers are an instruction that permits counting a specific time while the input connected to it is activated. Moreover, when the time is over, this timer will be energized and, an input with this timer address can be used to turn on other instructions. As can be seen in Figure 2.14, timers are added, for instance, as TIM T0 #800, where T0 is the timer number (T0, T1, T2, T3…) and #800 is referred to as the millisecond.

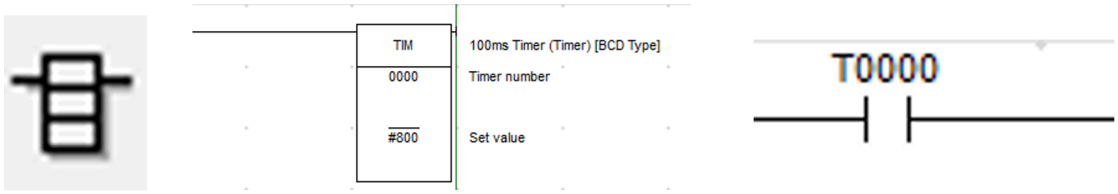

Fig. 2.14. Timer instruction

- **Reversible counter**

A reversible counter is a counter command with the aim of adding up or down a set value. It has three connections, the upper one increases by one meanwhile the middle one, decreases by one. The other one is a reset. As can be shown in Figure 2.15, a reversible counter can be set as New Instruction – Find Instruction – CNTR(012). The first number is the counter address and the other refers to the set value.

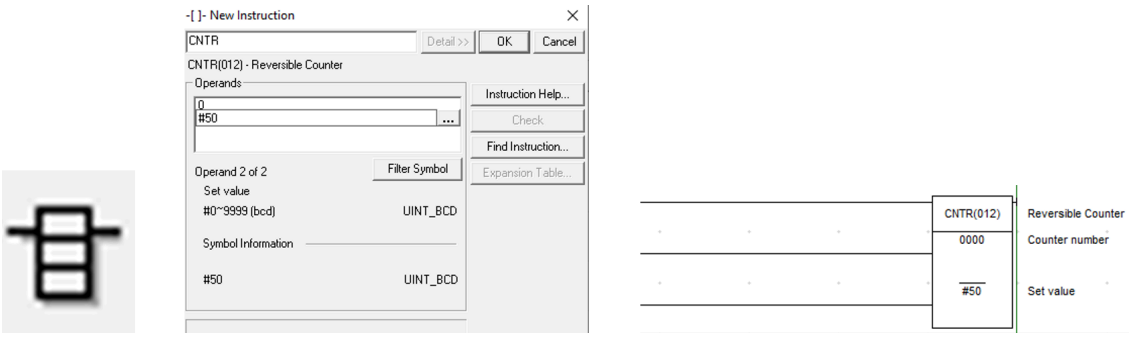

Fig. 2.15. Reversible counter instruction

On the other hand, a normal counter, which only counts either up or down, can be configured too.

- **Differentiate UP/DOWN inputs**

Basically, these differentiated inputs allow us to turn on a particular output momentarily, so this output will be no longer activated. Differentiate UP inputs will be energised when these inputs go from OFF to ON, contrary to differentiate DOWN inputs. This command is commonly used for "Push-button" switches (Figure 2.16).

Fig. 2.16. Differentiate UP/DOWN inputs

- **Keep function**

Keep functions operate as the relay latches, latching relay, which is set by S and reset by R. When S is in the ON state, operating instructions specified output will remain in a state of ON and ON until reset, regardless of whether S is ON or OFF. Frequently utilized to keep the ON state in differentiated inputs, as shown in Figure 2.17.

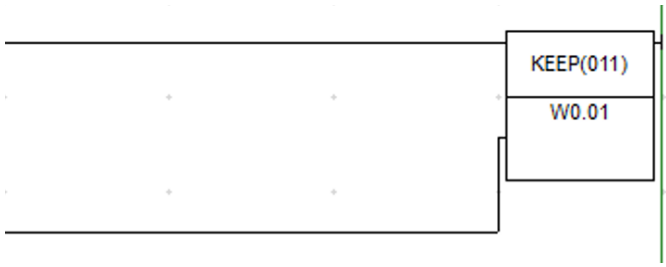

Fig. 2.17. Keep instruction

- **Move function**

The move function is used to move a value from one memory location to another. It is commonly used in ladder logic programming to transfer values between data registers, timers, counters, and other memory locations. It can also be used to copy values from input devices, such as switches or sensors, to output devices, such as motors or lights. "Source" represents the memory location from which the value is to be moved, and "Destination" represents the memory location where the value is to be moved (Figure 2.18).

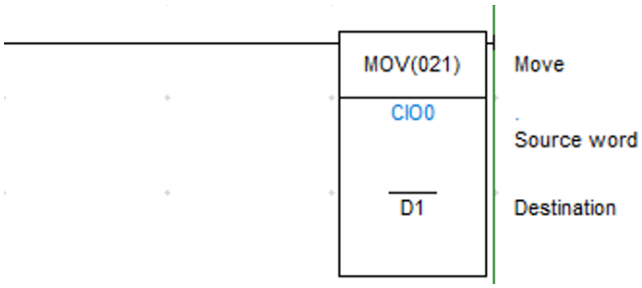

Fig. 2.18. Move instruction

- **Comparison data function**

Comparison instructions in PLC are used to test pairs of values to condition the logical continuity of a rung. As an example, shown in Figure 2.19, suppose an instruction is presented with two values. If the first value is less than the second, then the comparison instruction is true.

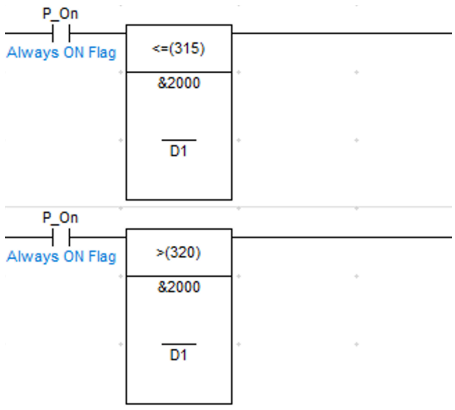

Fig. 2.19. Comparison data instruction

2.2. HMI

This thesis argues the creation of HMI software. It is a type of software used to control and monitor industrial processes and machines. It provides a GUI that allows users to interact with the machines and systems in a user-friendly and intuitive way.

2.2.1. NB-designer initialization

In my case, I will run the project in NB-DESIGNER, which is wholly integrated into the CX-One software suite. It is the programming software for many of Omron's HMI series.

Firstly, as can be seen in Figure 2.20, the user has to display the PLC and the HMI by selecting his own manufacturer. In this case, "HMI NB10W-TW01B" and "PLC OMRON CJ/CS/NJ". As we are connecting the setup via Ethernet, this must also be displayed. These are the IP addresses for the HMI and PLC distinctively [16].

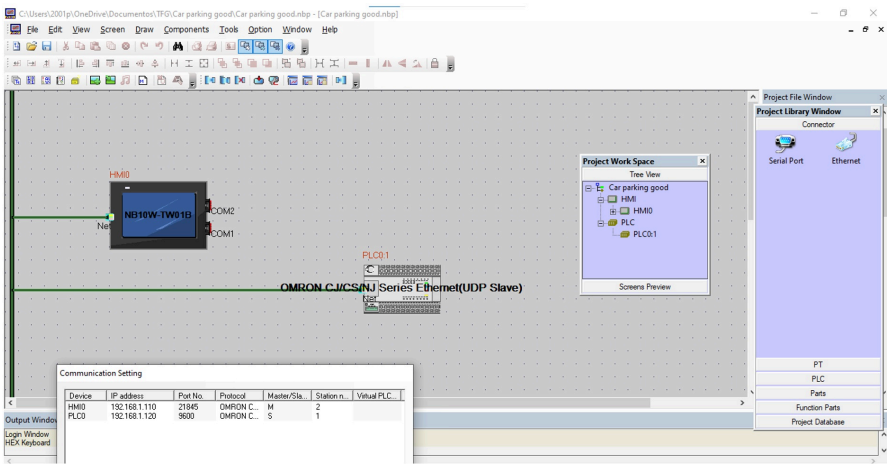

Fig. 2.20. Screen where communication between HMI and PLC is set

On the other hand, we should set the IP addresses and Port No of the PLC and HMI, including the HMI Subnet Mas (Figure 2.21).

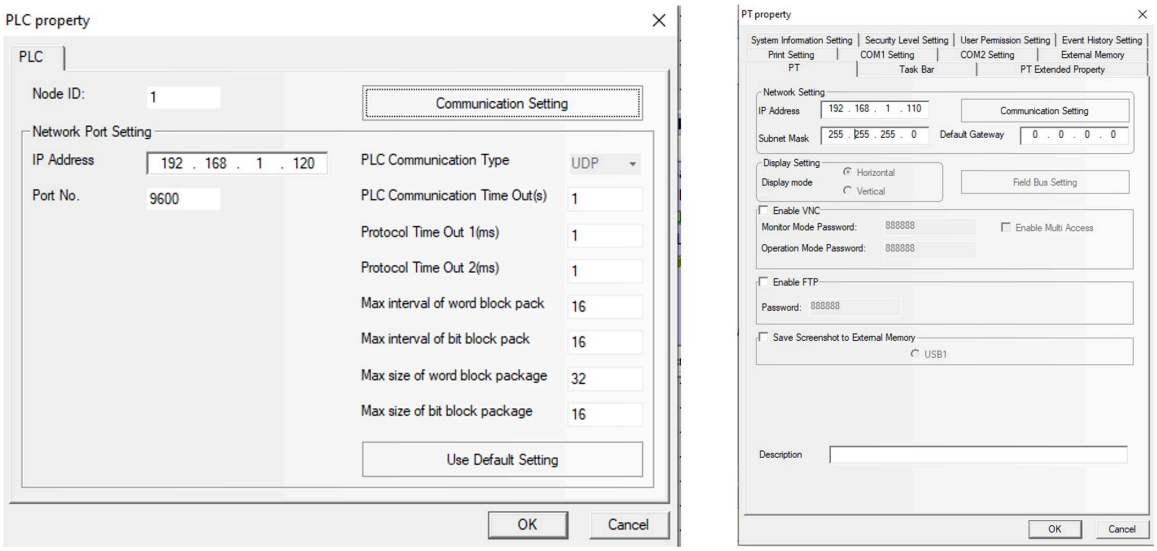

Fig. 2.21. IP address and Port No setting

Eventually, by pressing “HMI0” within the “Screens Preview”, we will be ready to start working. The program can be tested either in simulation or in the HMI. It is required to compile the program and afterward, if it has been successful, download it (transfer to HMI) or through “Offline Test”. In Figure 2.22 can be shown the required settings in order to transfer the program to the HMI.

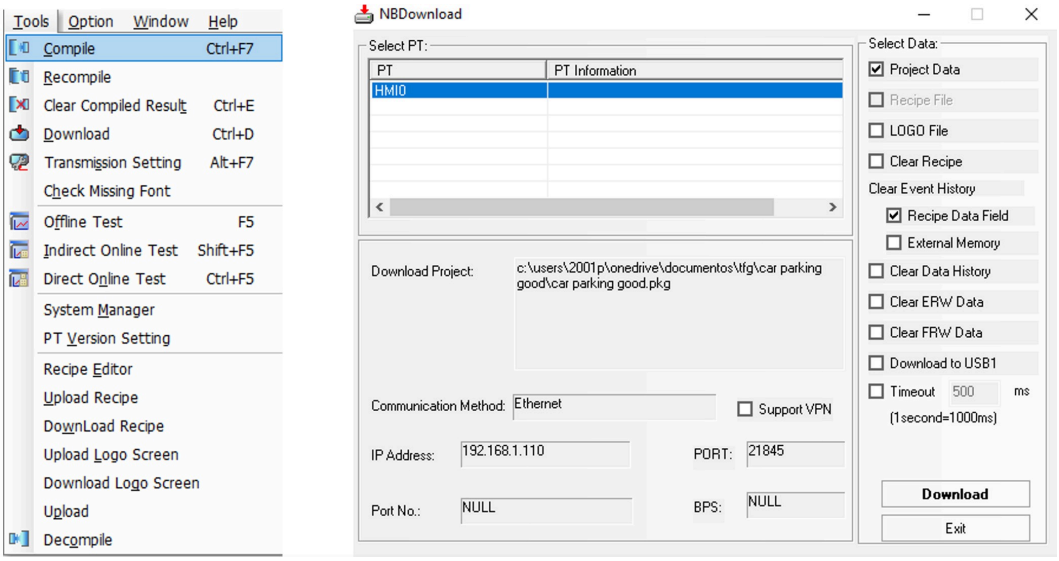

Fig. 2.22. Program compilation

2.2.2. NB-Designer commands

After finishing with all the laborious initialization, the basic commands and functions will be explained. As we can see in Figure 2.23, this is how the program looks like.

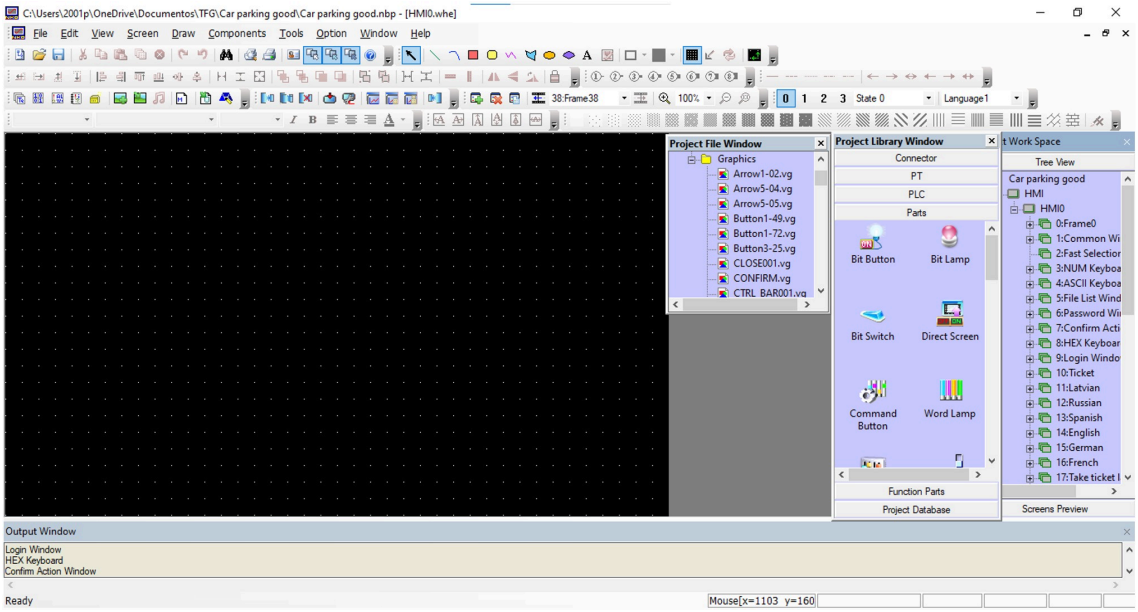

Fig. 2.23. NB-Designer main screen

On the right, “Screens Preview”, all the screens are displayed. It can be seen that from 0 to 9, including both screens, are related to preconfigured screens. Most likely we will add new screens instead of using those, however in some programs is also common to use screen 0, “Frame0”.

Firstly, in order to add a new screen, the user should press the right button on “HMI0”, and then, add and configure the new screen image. This screen can be designed within “properties”, like changing its name or background color (Figure 2.24).

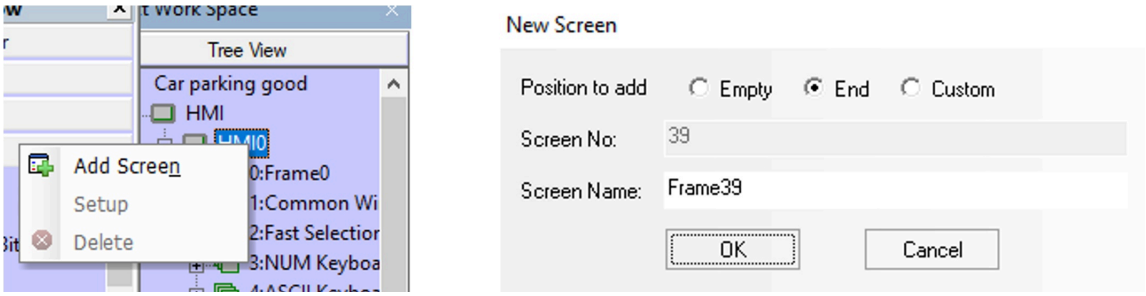

Fig. 2.24. Adding a new screen

Secondly, to create a title or add notes, several rectangles with some text inside can be added from the menu bar on the top figure (Figure 2.25).

Fig. 2.25. Adding text

When the user has created several screens, perhaps is wondering how to change from one screen to another within the simulation. So, the programmer can add the button “Function key”, select therefore which screen wants to appear on by pressing this button (Figure 2.26).

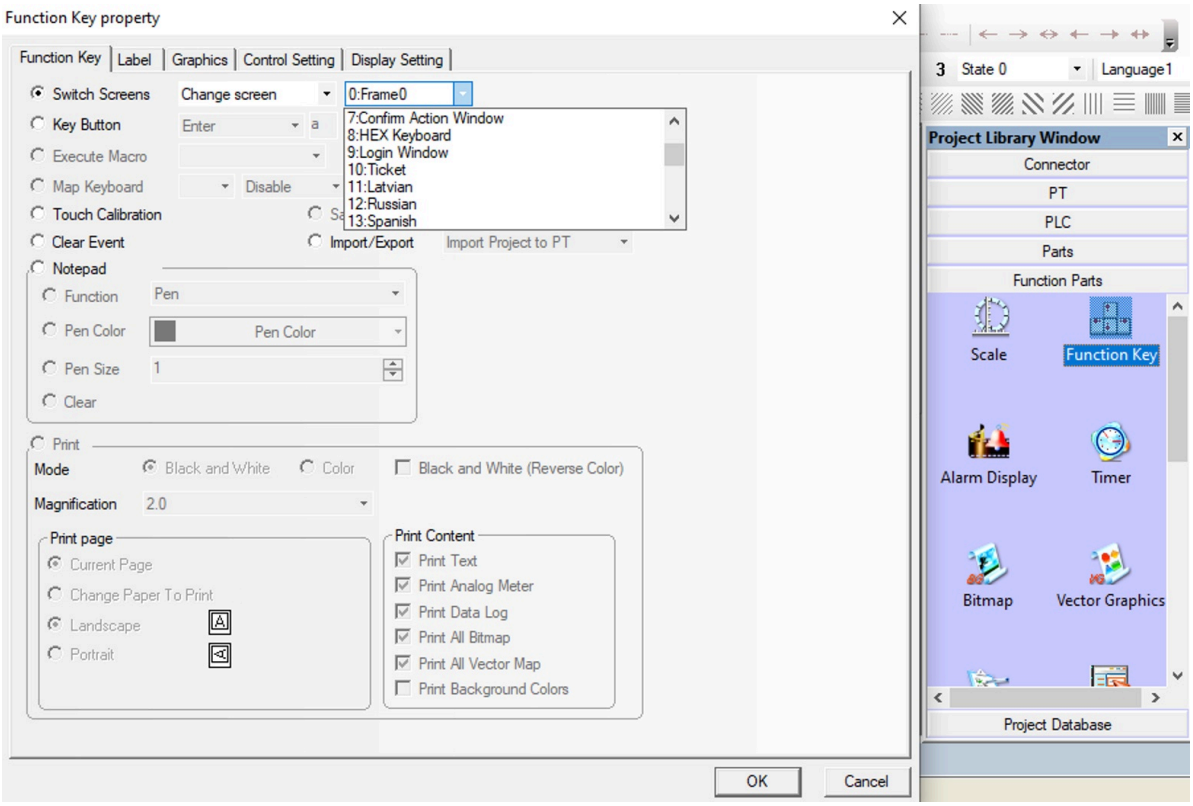

Fig. 2.26. Function Key property

On the other hand, with the aim of selecting which screen will be the initial, it can be set by pressing HMI0 and choosing the screen that suits better, as seen in Figure 2.27.

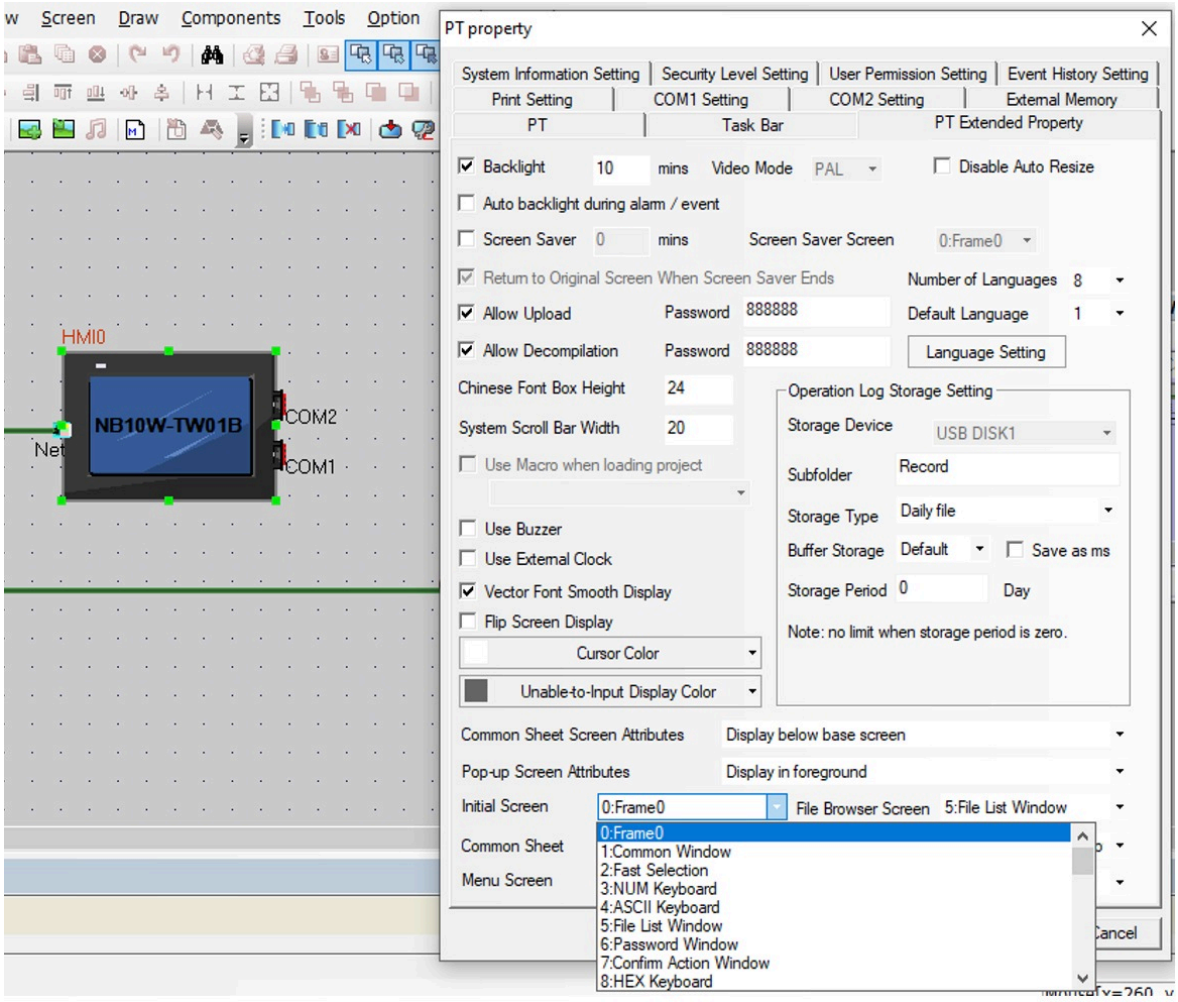

Fig. 2.27. PT property

Regarding input and output buttons, can be added within “Parts”. Basically, when the developer wants to add an input button, he can choose between “Bit Button” or “Bit Switch”, both are quite similar. These input buttons have the same function as CX programmer and PLC inputs. As a result, they have the possibility to be linked with the PLC and the CX-programmer software by setting the same address. This means that activating this kind of input button in the HMI screen will also activate a PLC input and a ladder diagram input bit, and viceversa. If the user has in mind making a program only involving the HMI, the variable LB with whatsoever address should be set (Figure 2.28).

On the contrary, with the aim of linking ladder diagram W input bits with the HMI inputs, it can be configured by setting the W_bit variable and the same address as the bit in the ladder software, as seen in Figure 2.29.

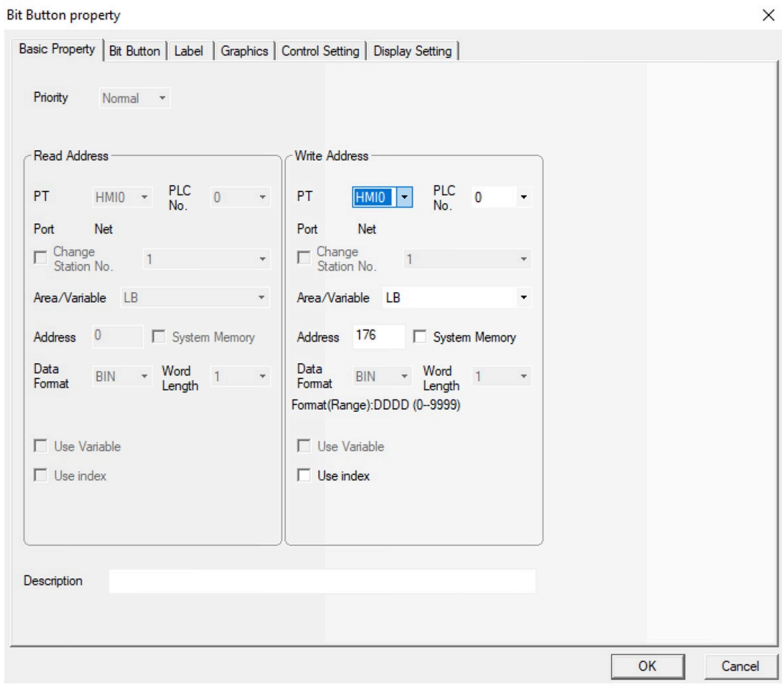

Fig. 2.28. Bit Button property

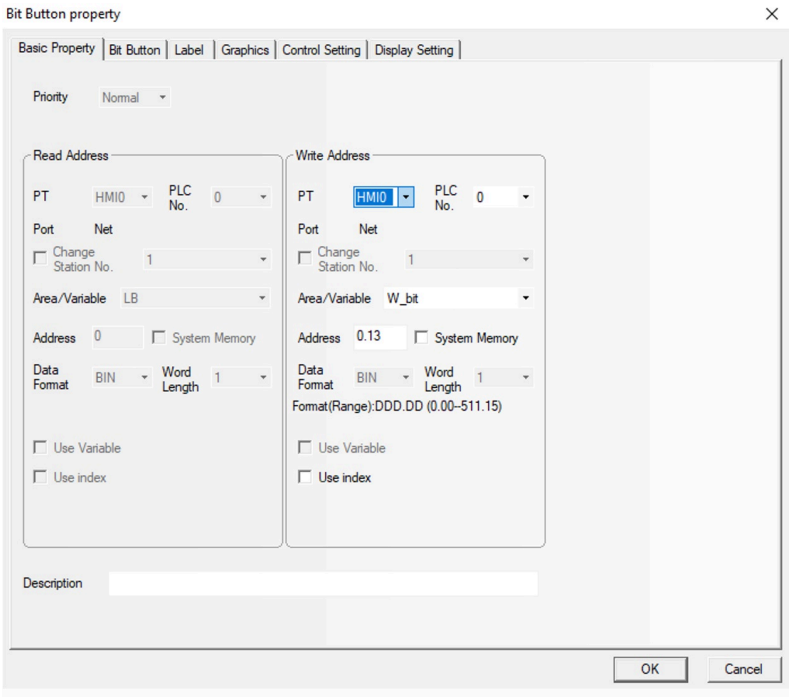

Fig. 2.29. Bit button property

Continuing with the input properties, the input button can be configured as a set, reset, alternate, or momentary. Generally, the set is referred to exclusively turn on a bit, and the

reset function is solely to turn it off. Otherwise, the alternate function will turn on/off the bit when pressing the button, and the momentary will work as a switch, the bit will be activated while the button is being pressed (Figure 2.30).

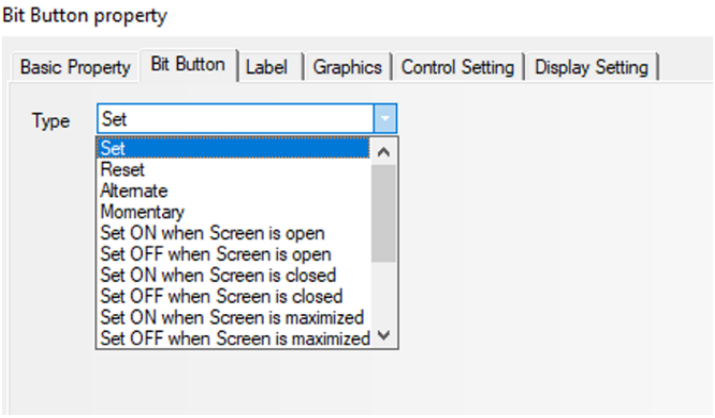

Fig. 2.30. Bit Button property

Moreover, to select a symbol for each state, as seen in Figure 2.31, can be chosen within the NB-Designer graphics or imported from the internet. The last option allows the user to design specific buttons.

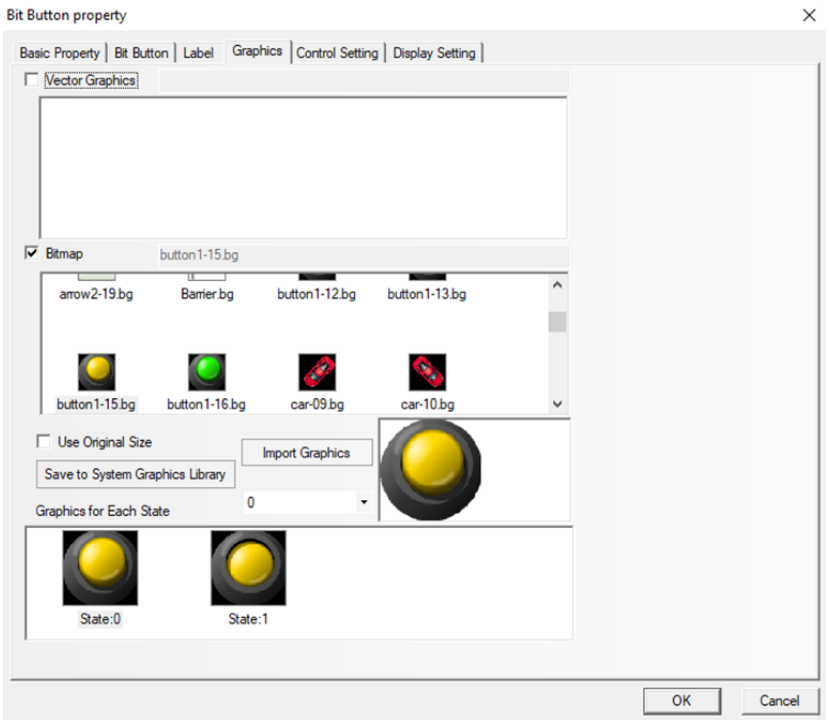

Fig. 2.31. Bit Button property

Later on, the outputs buttons are commonly used to visualize the inputs interactions. As a result, an output can be energised or de-energised by turning on manually an input button in the HMI. These output buttons can be found in the program as “Bit lamp”. In addition, they have the possibility to be linked with the PLC and the CX-programmer software by setting the same address. This means that when a PLC output or ladder diagram output bit is activated, these HMI output buttons will be activated too.

In Figure 2.32 it can be shown that the switch 0.05, attached to the PLC, has been linked with an HMI output button. So, the action of activating this PLC switch will be shown in the HMI output button.

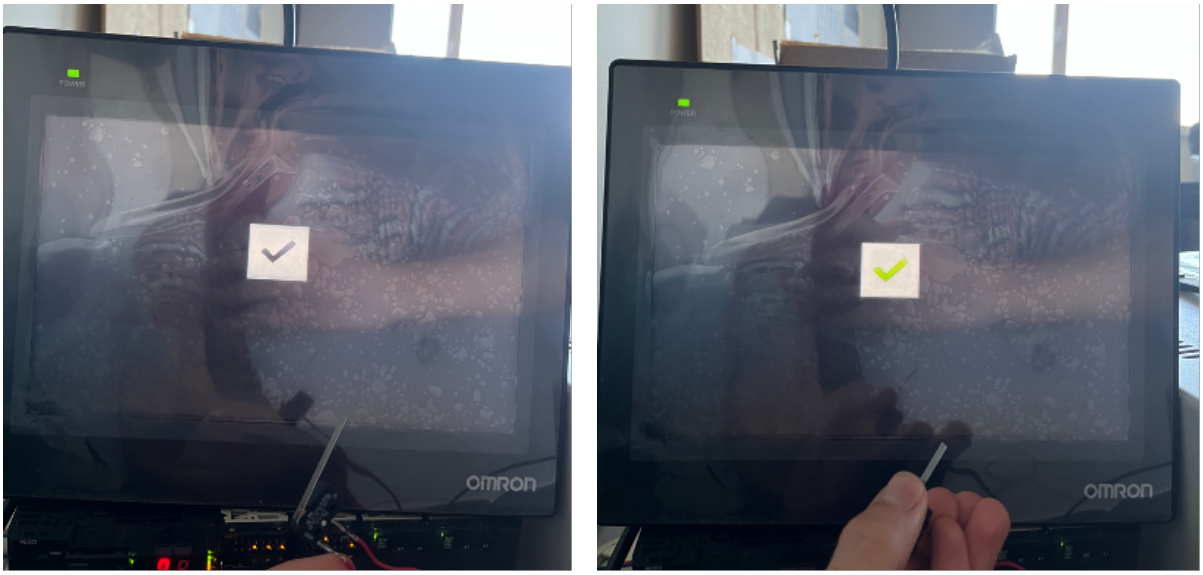

Fig. 2.32. Turning on a HMI output button by pressing a switch attached to the PLC

Overall, the output configuration is practically the same as the inputs buttons, however, in Figure 2.33 it can be seen that there are different types of bit lamps. This fact can help to hide some outputs while they are not turned on, and vice-versa. Typically use in the automated parking program to hide the car and barrier images.

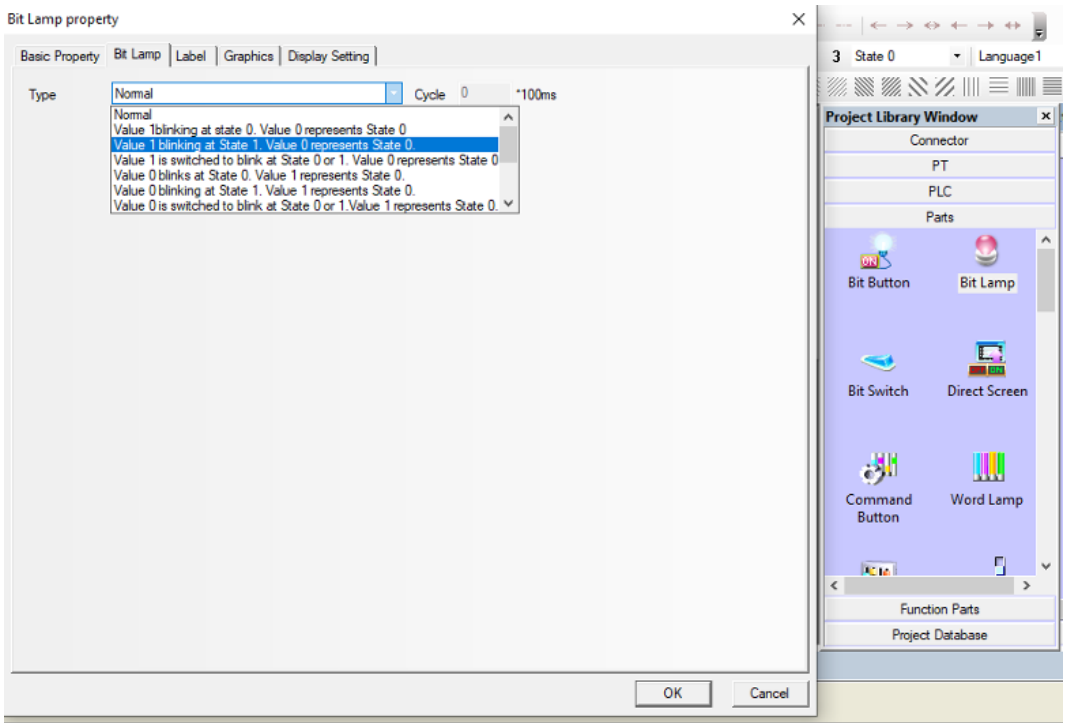

Fig. 2.33. Bit Lamp property

As an example of all of that, in Figures 2.34 and 2.35 it can be seen an input with the address “LB 120”, which is going to turn on an output with the same address.

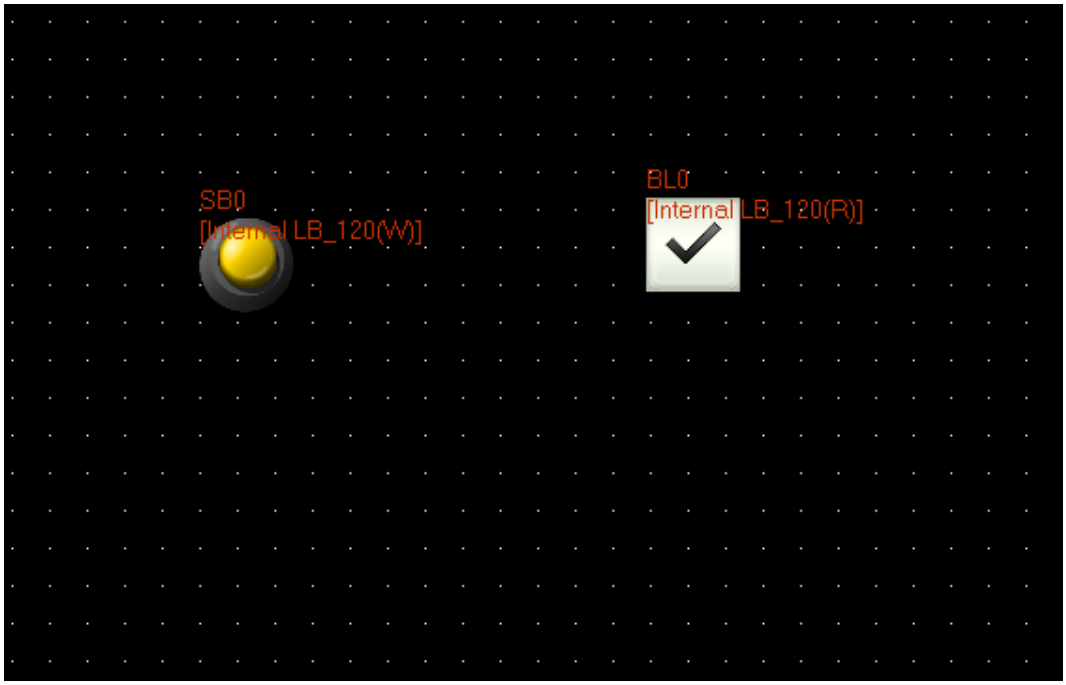

Fig. 2.34. Example of an input and output with the same address

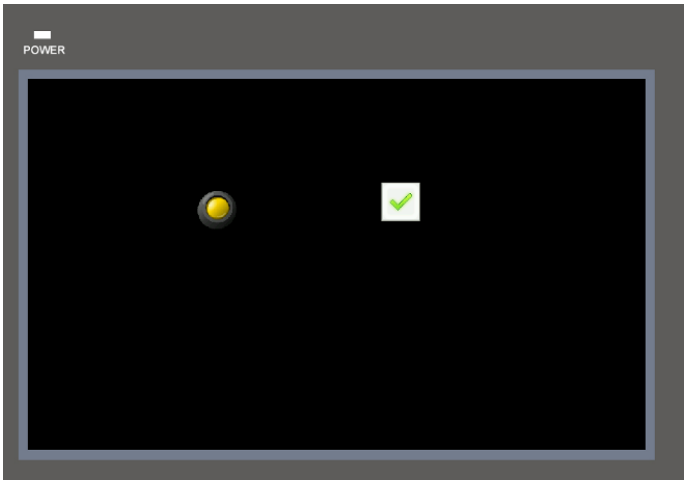

Fig. 2.35. Example of turning on an output by pressing the input button

Additionally, using the command “Number display”, has the possibility to show visually a number in the HMI screen. This value can be related to a PLC counter, PLC timer, analog output… If the user has the intention to link it with some analog input, variable “D” with its address must be configured. Utilizing commonly “BIN” as data format. On the other hand, variables “C” and “T” should be used for counters and timers, respectively. Also, they commonly operate in the data format “BCD”. It can be shown in Figure 2.36.

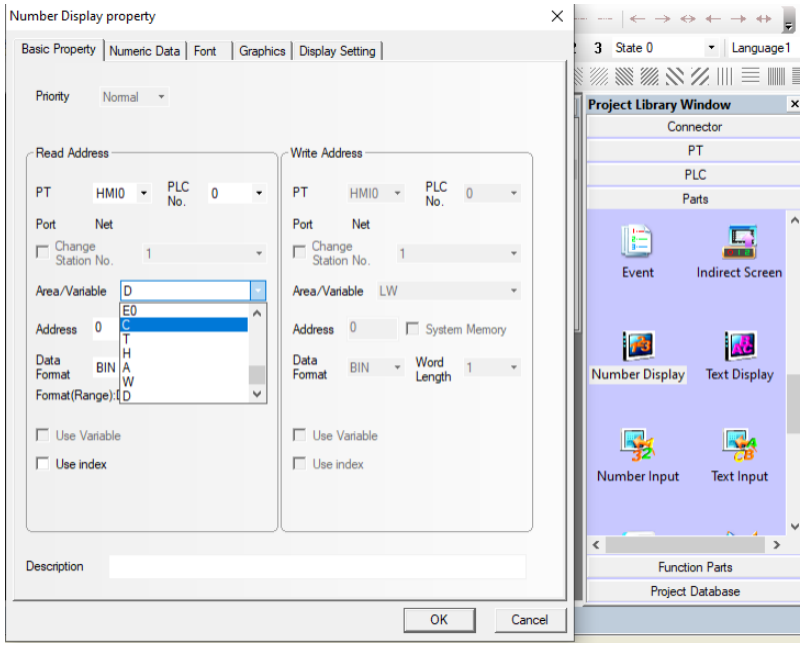

Fig. 2.36. Number Display property

In Figure 2.37, we can see the possibility to choose the storage format. Also, a minimum and maximum value can be configured, which can help to display an analog input.

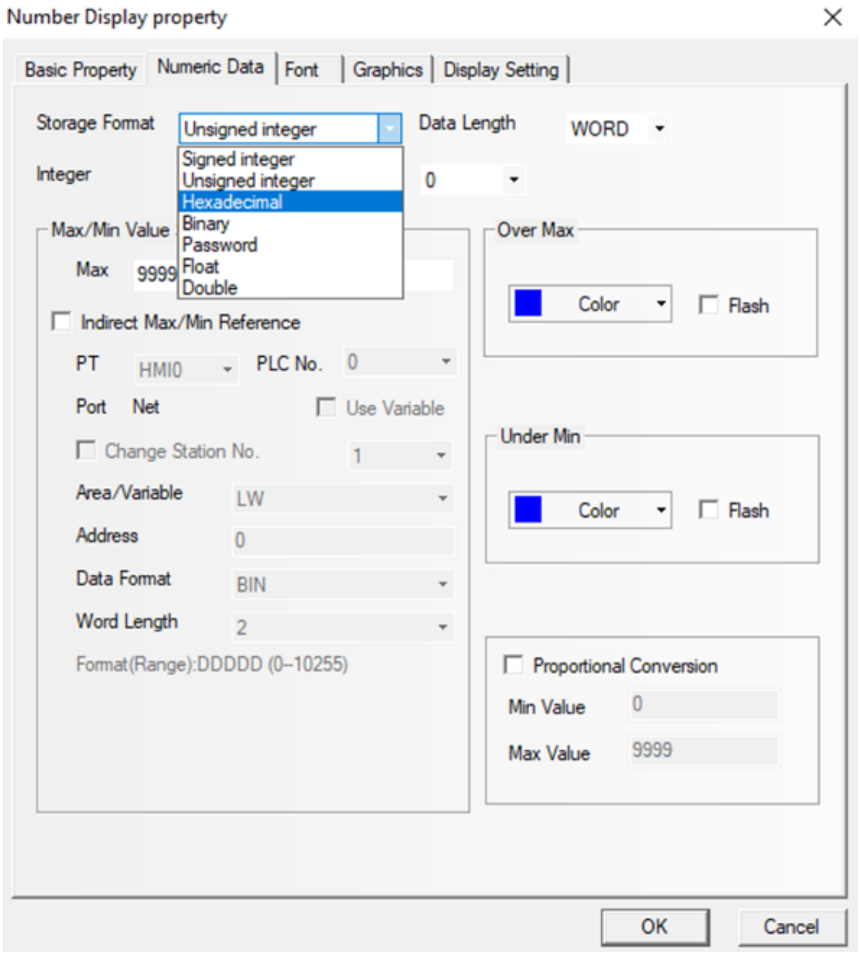

Fig. 2.37. Number Display property

3. EXAMPLES

3.1. Digital input

First off, in order to turn on a PLC digital input, it is necessary to display an addressed input in the ladder diagram software, as seen in Figure 3.1, and link it with some other commands. In this example, an input has been linked with an output [17].

Fig. 3.1. Input linked with an output by ladder logic

Then, as shown in Figure 3.2, we can turn it on by activating the switch “0.01” attached to the PLC or directly activating the input from the ladder program.

Fig. 3.2. Turning on the 0.01 input

This fact leads us to turn on the led 1 on the digital input rack, as shown in Figure 3.3.

Fig. 3.3. Turning on the LED 1

3.2. Digital output

In order to turn on a PLC digital output, it is required to display an input with an address in the ladder diagram software, as seen in Figure 3.4. Afterward, this output should be linked to an input, and thus, when the input is on, the output will be too.

Fig. 3.4. Turning on the 1.04 output

This fact leads us to turn on the led 4 on the output rack, shown in Figure 3.5. In addition, by attaching an external component to the pin 4, this device will be also activated. For instance, if a motor is attached, will rotate while this pin is activated.

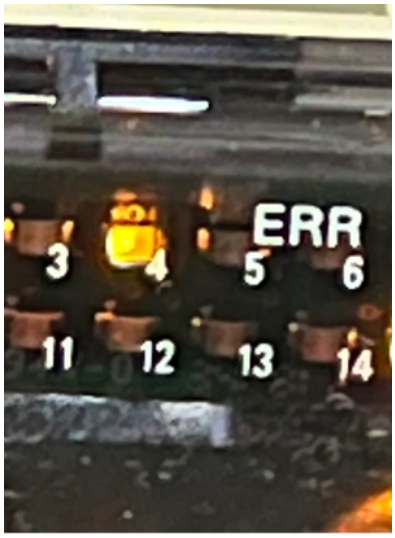

Fig. 3.5. Turning on the LED 4

Later on, once the user has improved his PLC programming skills, he might be able to program big rungs grouping together different inputs, outputs, and commands. So, many LEDs can be involved, and hardware such as switches, motors…

3.3. Analog input

An analog resistor has been implemented as an example of analog input. So basically, by connecting the wire in specific bits, we will be able to regulate the input voltage between 0-10V. As seen in Figure 3.6, with “MOV(021)” we will be able to set the input voltage [18].

Fig. 3.6. Example of analog input with MOV(021) instruction

As a result, we will be able to achieve a decimal and hexadecimal value by changing the analog resistor value attached to the PLC. These numbers are directly related to the voltage. Once we have got the range of values, by using a "Comparison" we will manage to set a threshold between the low voltage and high voltage (Figure 3.7). These outputs will be implemented in the parking garage program.

Fig. 3.7. Example of Analog Input with Comparison Instruction

On the other hand, depending on the voltage and the input number, the input conversion value and conversion data holding addresses will change. As seen in Figure 3.8, there are different input values range depending on the bits where the component is attached. Moreover, it might be some type of error referred to the hexadecimal and decimal voltage range, which means that these range of numbers might be slightly lessened.

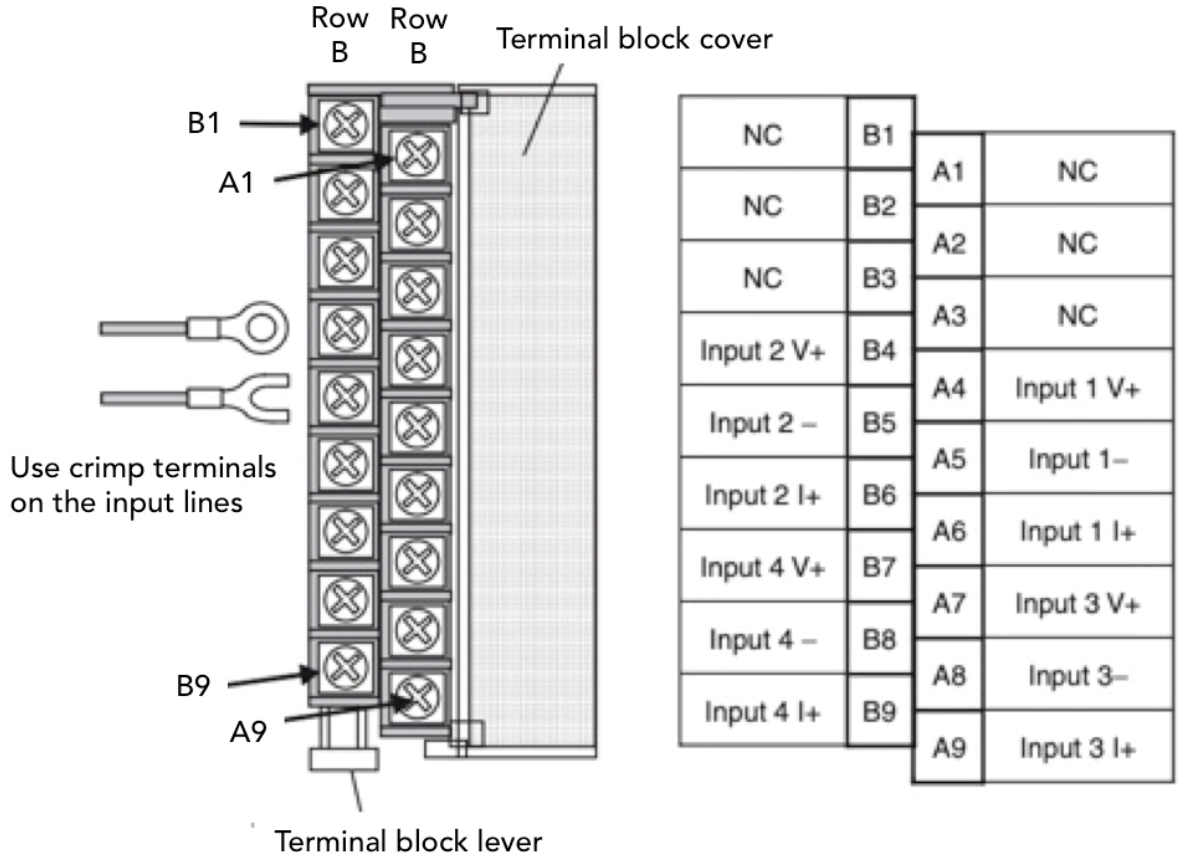

Fig. 3.8. Analog Input Terminal Block Arrangement

3.4. Analog output

A multimeter might be implemented as an example of analog output. Also with "MOV(021)" we might be able to set the output voltage.

As a result, we will be able to introduce a decimal value either in the ladder diagram or by changing the analog resistor value attached to the PLC, which also will be converted to hexadecimal. So, the multimeter will display the voltage set through the analog resistor or ladder diagram.

On the other hand, depending on the voltage and the output number, the output conversion value and conversion data holding addresses will change. There are different output values range depending on the bits where the component is attached. Moreover, it might be some type of error referred to the hexadecimal and decimal voltage range, which means that these range of numbers might be slightly lessened.

3.5. Parking garage

This thesis argues a demonstration test bench based on the hardware and software explained before. So, a real example that includes the PLC, HMI, computer, and external components will be explained. This real example can be tested separately with the PLC, HMI, or computer, however, it can be also tested by combining two of them or even three.

- The program consists of a parking garage simulation where basically cars enter or exit.
- Consists of two sensors, two ticket machines, and the barrier.
- In the entrance process, the car has to take the entrance ticket in order to enter the parking garage. Then, the car will go towards the barrier, and sensor 2 will detect it and will make the barrier up. Later on, the car will go through the barrier and after that, sensor 1 will detect the car getting in and will make the barrier down.
- In the exit process, the car has to return the entrance ticket in order to get out of the parking garage. Then, the car will go towards the barrier, and sensor 1 will detect it and will make the barrier up. Later on, the car will go through the barrier and afterward, sensor 2 will detect the car getting out and will make the barrier down.
- In case the car finally regrets going through the barrier after being detected by the sensor, an 80 seconds timer security has been implemented in order to make the barrier down automatically.
- The barrier needs different voltages in order to go either up or down. It will need a low voltage to make the barrier down and will be necessary a high voltage in order to make the barrier up.

3.5.1. Data flow diagram

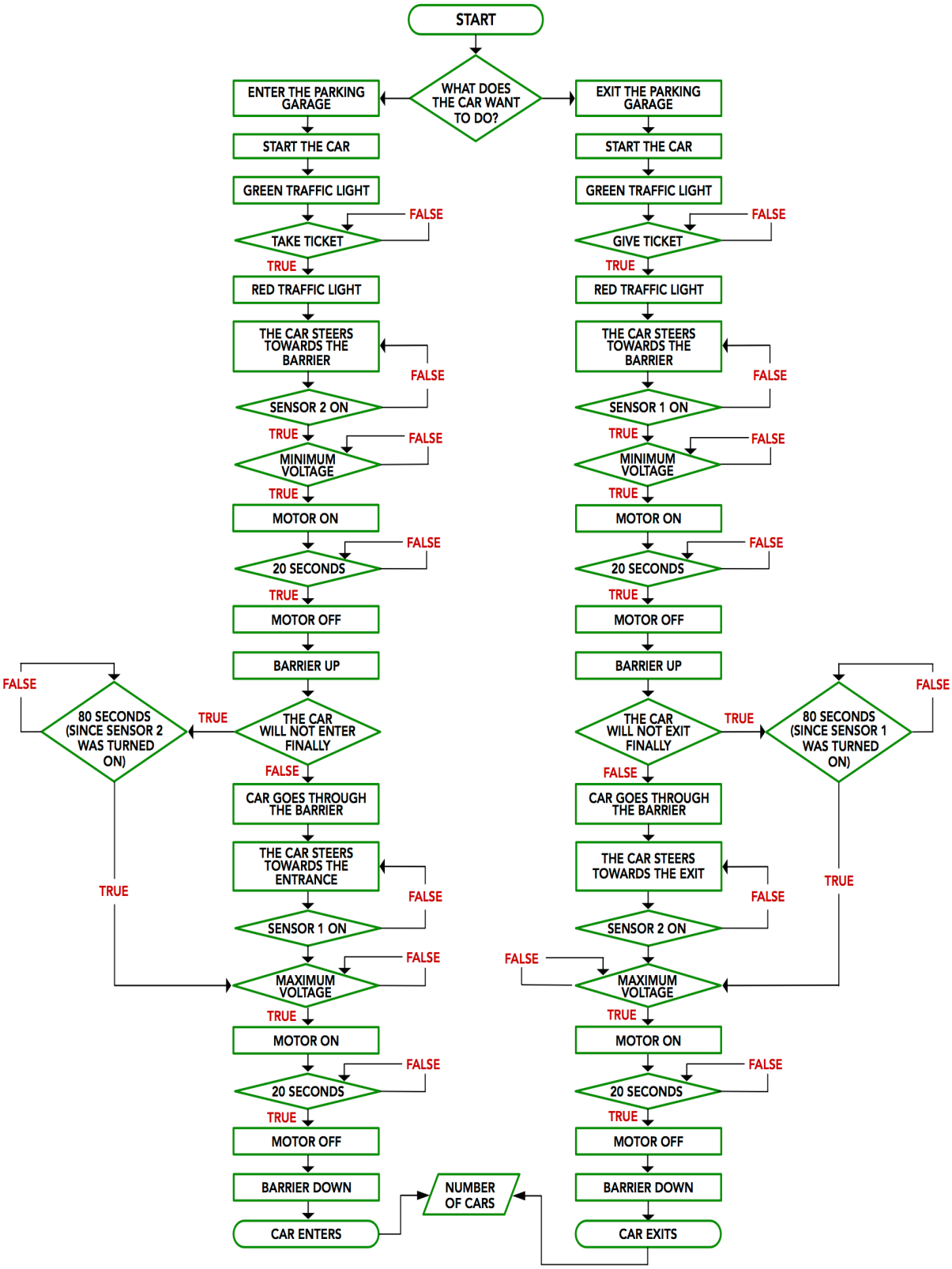

Fig. 3.9. Data flow diagram

3.5.2. Hardware Setup

3.5.2.1. Components

Figure 3.10 shows the different components that have been taken into account.

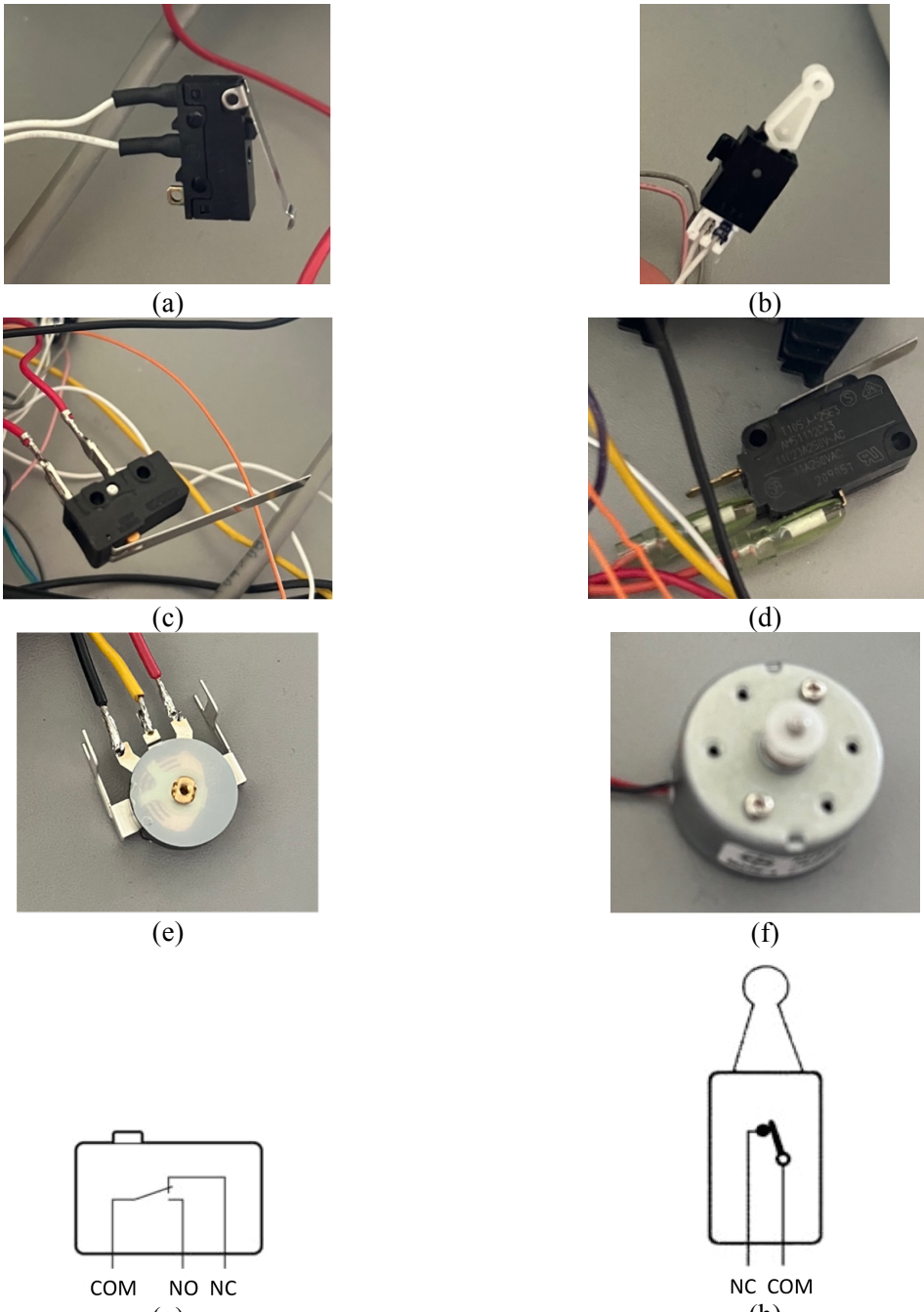

Fig. 3.10. (a) Switch 0.11 digital input “Take ticket button”. (b) Switch 0.01 digital input “Give ticket button”. (c) Switch 0.05 digital input bit “Sensor 2”. (d) Switch 0.09 digital input bit “Sensor 1”. (e) Analog resistor 0-10V “Barrier voltage”. (f) DC motor 1.10 and 1.12 digital output bits “Barrier”. (g) Contact Form of the Switch Differentiated up. (h) Contact Form of the Switch Differentiated down

3.5.2.2. Motor functioning

The “H-bridge” electronic circuit has been set up to achieve the motor's rotation in both directions [19]. Basically, an H-bridge is a commonly used integrated circuit that can be used to control the direction of rotation of a DC motor. It is called an "H-bridge" because the arrangement of the switching elements looks like the letter H. Consists of four switches, which can be controlled independently to control the direction of current flow through the motor. To understand how the H-bridge works, imagine that you have a DC motor with two leads, labeled A and B. The H-bridge has four transistors, labeled VT1, VT2, VT3, and VT4. When a voltage is applied to VT1 and VT4 through the U1 optocoupler, current will flow from lead A of the motor to the ground, causing the motor to turn in clockwise. Conversely, when a voltage is applied to VT2 and VT3 through the U2 optocoupler, current will flow from lead B of the motor to the ground, causing the motor to turn anti-clockwise.

As it can be shown in Figure 3.11, X1 is related to pin 12 of the PLC digital output and X2 to pin 10. When they are activated, they will send 24V from the PLC to its respective optocoupler. On the other hand, X3 is referred to as the PLC ground. The circuit can be shown in Figure 3.12.

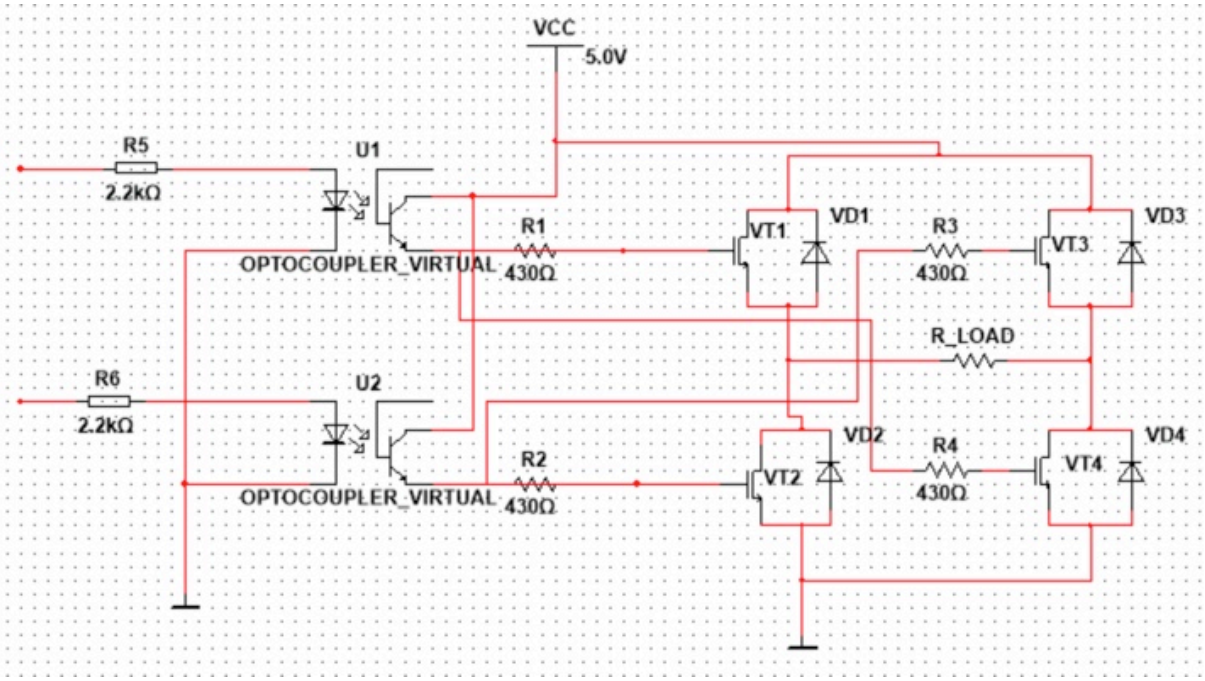

Fig. 3.11. H-Bridge circuit simulation

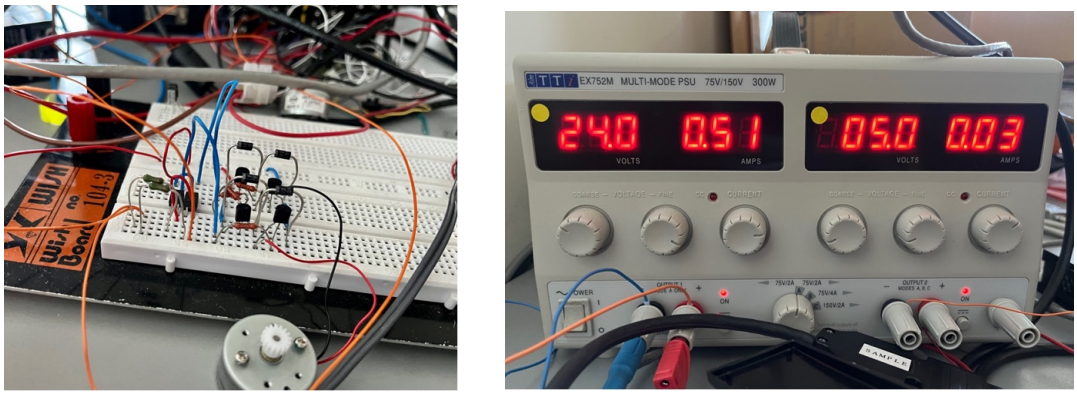

Fig. 3.12. H-Bridge circuit setup and voltage applied

In Figure 3.13, it can be shown when each component is either ON or OFF.

Fig. 3.13. ON/OFF diagram

3.5.3. Ladder diagram

In this section, I am explaining the ladder diagram software that I have developed for the automated parking garage process.

First of all, the car has to be started and ready to get into the parking garage. As we can see in Figure 3.14, the traffic light is green “1.03”, which allows the car to move forward. It will be always green if the red traffic light, address 1.02, is OFF. In the PLC, pin 3 located on the output rack, will be illuminated.

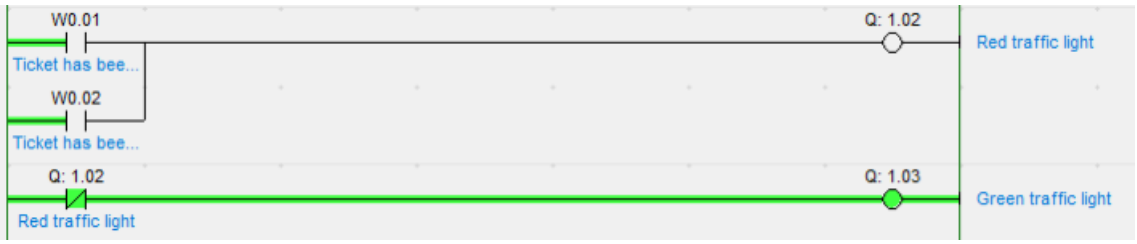

Fig. 3.14. Green traffic light, cars are allowed to continue

After that, related to the entrance, the car must take a ticket to enter the parking garage. So, there is attached to the PLC a switch that simulates the entrance ticket machine. This switch, address 0.01, is differentiated UP and will activate bit W0.01, which means that the ticket has been taken. In addition, the software will be reset when T2 is turned on, referring to the time while the barrier is going down, and hence, the process will be completed, as it will be shown below (Figure 3.15).

The pin 1 on the input rack will be temporarily illuminated, as a result of turning on the switch 0.01.

On the other hand, the W0.13 bit, “HMI take the ticket”, and the W0.09 bit, “stop the process”, are related to the HMI simulation.

Fig. 3.15. The car takes the entrance ticket in order to start the process

Once the ticket has been taken, the traffic light will be automatically red, which will not permit other cars neither enter nor exit (Figure 3.16). In the PLC, pin 2 located on the output rack, will be illuminated.

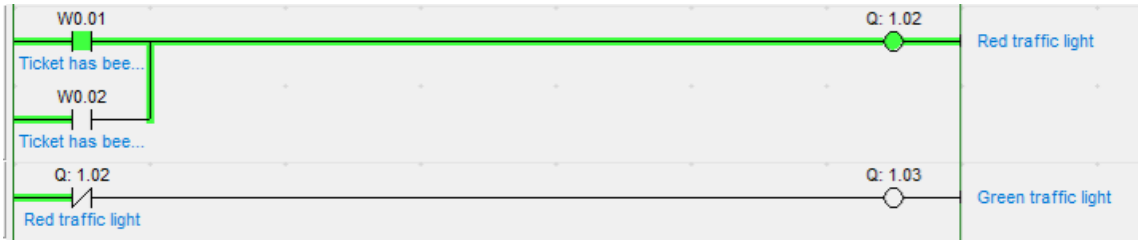

Fig. 3.16. Traffic light turn into red, cars are not allowed to continue

Later, the car steers towards the barrier, and a sensor detects it. A switch differentiated UP attached to the PLC will simulate this sensor. Furthermore, this sensor 2, address 0.05, will energise the bit W0.00, referred to as the entrance mode. Also, T2 appears with the same performance as we have seen before (Figure 3.17).

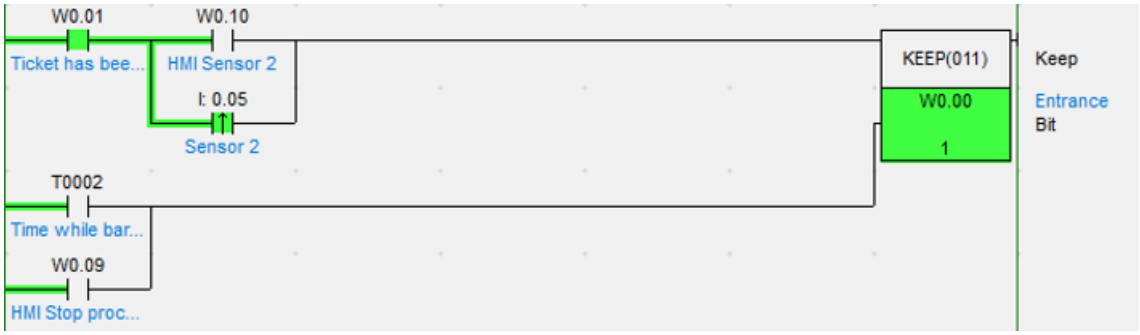

Fig. 3.17. Sensor 2 detects the car going toward the barrier

The pin 5 on the input rack will be temporarily illuminated, as a result of turning on the switch addressed as 0.05.

On the other hand, the W0.10 bit is the simulation of sensor 2 in HMI programming.

Once Entrance mode is activated, an eighty seconds security timer will start running. In case no car enters through the barrier after this time, the barrier will be automatically down. Therefore, if the car somehow regrets and decides not entering into the parking garage anymore, this timer allows certainly lower the barrier and finish the process. It can be shown in Figure 3.18.

Fig. 3.18. Eighty seconds security timer starts running

The pin 0 on the output rack will be illuminated during this time, addressed as 1.00. It would be the same process if the car were exiting. In that case, the exit mode bit, W0.03 would be turned on, instead of the Entrance bit W.00.

At the same time, after activating the Entrance Mode, it can be shown in Figure 3.19 that sensor 2 and Entrance Mode both will activate bit W.0.04 “Barrier Up”. Then, the rung will be reset when T3 is turned on, referring to the time while the barrier is going up, and hence, the up process will be completed, as it will be shown below.

Fig. 3.19. Barrier up mode is activated

Alternatively, if we were in the Exit mode, which I will explain afterward, bit W0.03, referred to as Exit mode, and bit 0.09, related to the sensor 1, would make bit W0.04 “Barrier UP” energised.

On the other hand, the W0.10 and W0.11 bits both are the simulation of the sensors in HMI programming.

In this stage, it is necessary to make the voltage high, so the barrier will be ready to go up. As a result, a twenty seconds timer will start to count and a motor attached to the PLC, as an output, will rotate in a clockwise direction. The barrier will be rising while T3 is running. When T3 is over, the barrier will be totally up, and also the motor will stop rotating. It can be shown in Figure 3.20. After all, the barrier will be completely up, and the car will go through it.

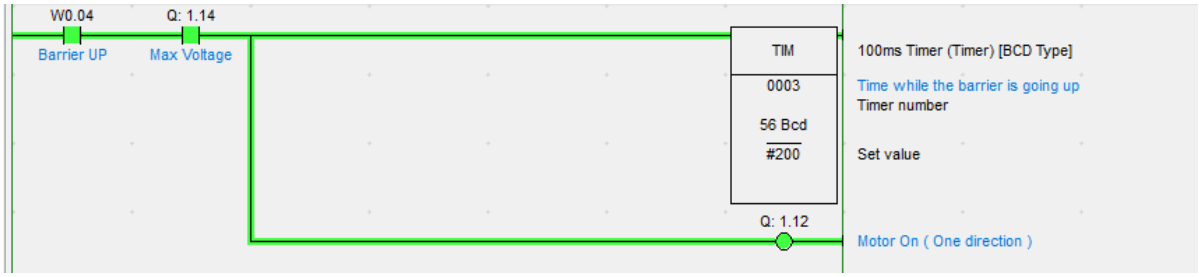

Fig. 3.20. The barrier is raising and the motor rotating

Consequently, when the car went through the barrier, it will steer towards the entrance. At this point, a sensor will detect the car driving away, simulated by a switch differentiated UP. Furthermore, the entrance mode bit and this sensor 1, address 0.09, will energise the bit W0.05, referred to as the barrier down mode. Also appears T2 with the same performance as we have seen before, Figure 3.21.

Fig. 3.21. Sensor 1 detects the car after it has gone through the barrier, and “Barrier Down” mode is activated

Alternatively, if we were in the Exit mode, which I will explain afterward, bit W0.03, referred to as Exit mode, and bit 0.05, related to sensor 2, would make bit W0.05 “Barrier Down” energised.

On the other hand, the W0.10 and W0.11 bits both are the simulation of the sensors in HMI programming.

Currently, it is necessary to make the voltage low, so the barrier will be ready to go down. As a result, a twenty seconds timer will start to count and a motor attached to the PLC, as an output, will rotate in a counterclockwise direction. When T2 is over, the barrier will be down and also the motor will stop rotating. It can be shown in Figure 3.22. After all, the barrier will be entirely down, and the process will be totally finished and reset.

Fig. 3.22. Barrier is going down and the motor rotating

However, speaking about the case when no car enters, which we have discussed before. It has been implemented the bit W0.06 dubbed Security Barrier. So, we should take a look at Figure 3.23, when the security timer is done, T0. Therefore, when T0 is ON, and at once, T2 and W0.05 both are OFF, the Security Barrier down mode will be switched ON, W0.06. This command is reset when W0.01, “ticket has been taken” mode, is turned off again.

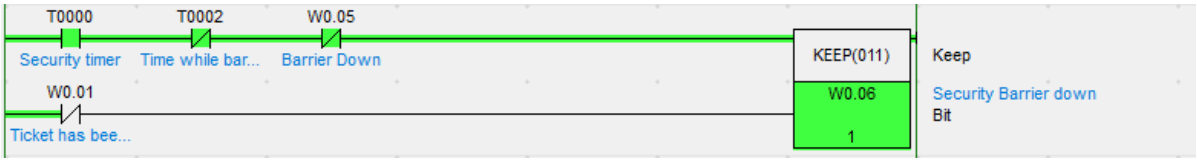

Fig. 3.23. “Security Barrier” settings

All of this means that, as no car has entered after the security time, it is supposed that the car has moved back. For that reason, it is required to make the barrier down and finish the process. Lately, the barrier will go down as explained before.

Additionally, with the aim of controlling how many cars there are within the parking garage, it has been added a counter. This counter will add up or down whether a car enters or exits, and thus, it will show the number of parking places available at the moment. In this example, the parking garage contains fifty parking places.

Basically, either the entrance mode is ON, W0.00, or the exit mode, W0.03, the counter will count in one way or another. Moreover, the bit "Barrier Down", W0.05, must be energised to claim that the car has truly gone through the barrier, shown in Figure 3.24.

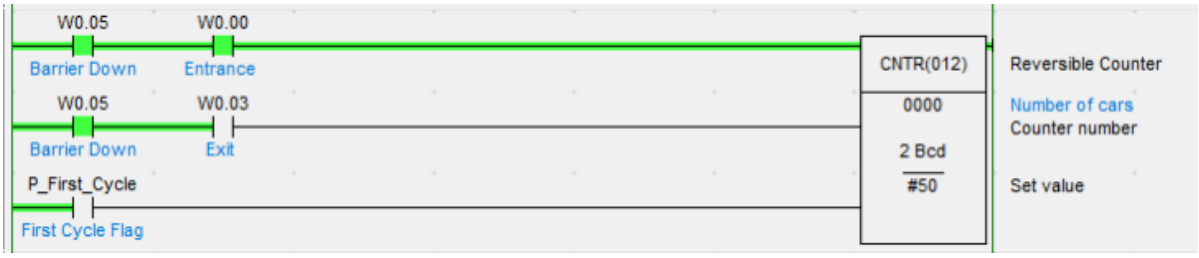

Fig. 3.24. Cars counter

After mainly explaining when the car enters the parking garage, when we refer to the exit mode, some slight differences have to be mentioned. In this case, the car must give the ticket that previously took entering the parking garage. So, a switch that simulates the exit ticket machine is attached to the PLC. Contrary to the "Take ticket" switch, this new switch with the address 0.11, is differentiated Down and will activate bit W0.02, which means that the ticket has been given. In addition, T2 appears with the same performance, as we have seen before. The pin 11 on the input rack will be temporarily illuminated due to turning on the switch 0.11, Figure 3.25.

On the other hand, the W0.14 bit, "HMI give the ticket", and the W0.09 bit, "stop the process", are related to the HMI simulation.

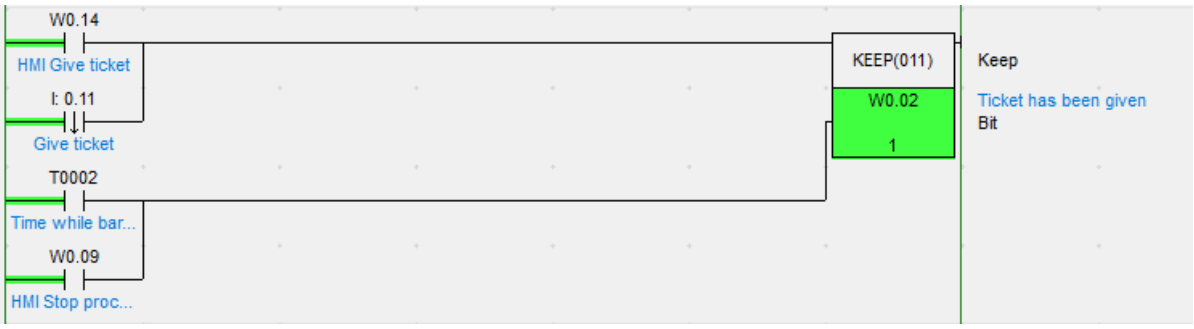

Fig. 3.25. The car gives the ticket in order to start the process

Later on, similar to the Entrance mode activation, sensor 1, addressed as 0.09, will turn on the exit Mode, bit W0.00. Also T2 appears with the same performance, seen previously. The pin 9 on the input rack will be temporarily illuminated due to turning on the switch addressed as 0.09, Figure 3.26.

On the other hand, the W0.11 bit is the simulation of the sensor in HMI programming.

Fig. 3.26. Sensor 1 detects the car going toward the barrier

3.5.4. HMI program

In this chapter, I am explaining the HMI software that I have developed for the automated parking garage process. The program comprises three areas (Figure 3.27), as follows:

- MANUAL SETTING
- PLC SETTING
- HMI SETTING

Fig. 3.27. Main screen

Speaking about the “Manual Setting”, the parking garage can be simulated manually by the user. This means that there will not be any interaction neither with the PLC nor with the computer, only the HMI screen will take part.

When the user will press the button, it will emerge another screen. Within this, exists the possibility to simulate either the ticket action or the entrance and exit process. Also, a screen with some cars from the parking and their number plate has been added. It can be shown in Figure 3.28.

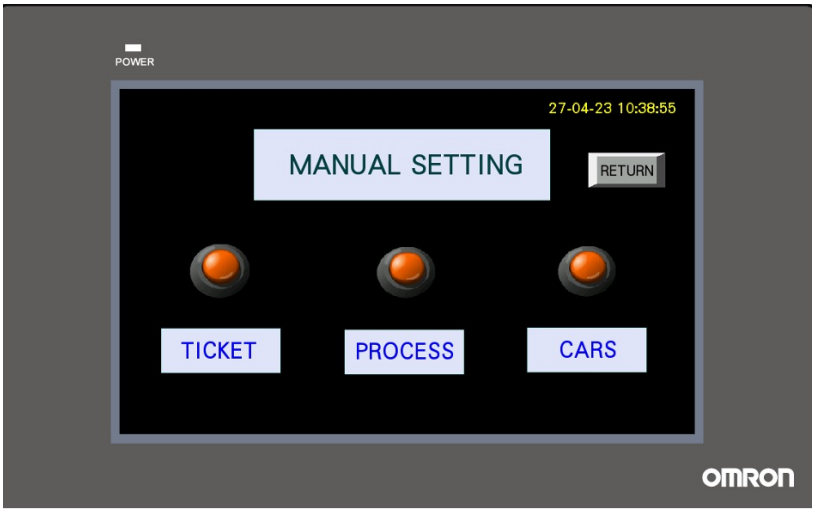

Fig. 3.28. Manual Setting screen

Appertaining to the ticket screen, firstly the user will be able to choose the ticket´s machine language. There is a huge variability of languages, shown in Figure 3.29, such as Latvian, Russian, Spanish, English, German, and French.

After the user has chosen the language which suits better for him, the user will be automatically redirected to the next step (Figure 3.30). In this stage, the customer should choose between taking or giving the ticket, as an entrance or exit process respectively.

Fig. 3.29. Select the language screen

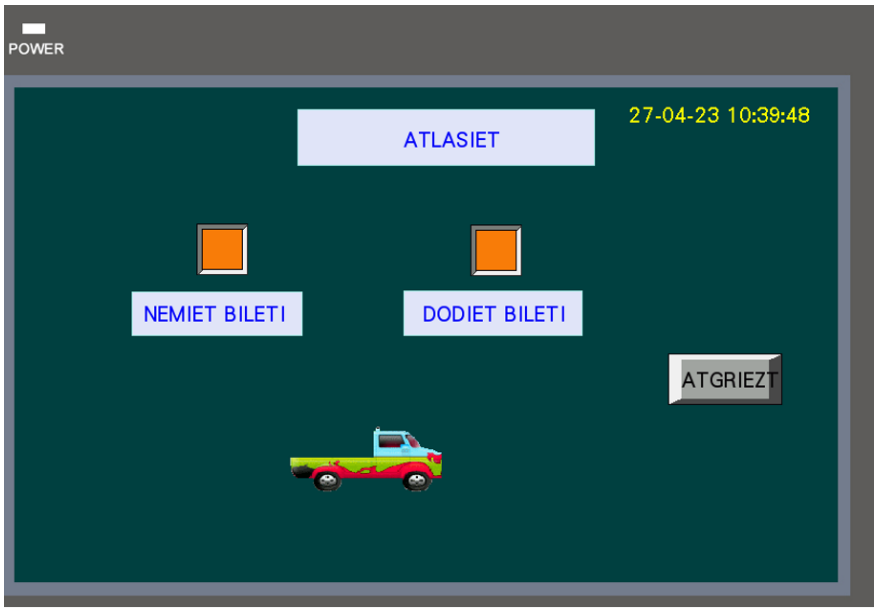

Fig. 3.30. Ticket process

Later on, if the customer selects “Take ticket”, another screen will come out. In this one, the buyer will be asked for paying 1,50 $ by card or cash. Then, after paying, the ticket will be ready to be taken and thus, the entrance process will continue. It can be seen Figure 3.31.

On the other hand, about giving the ticket to get out of the parking garage, the steps are practically the same, seen in Figure 3.32.

Fig. 3.31. “Take ticket process”

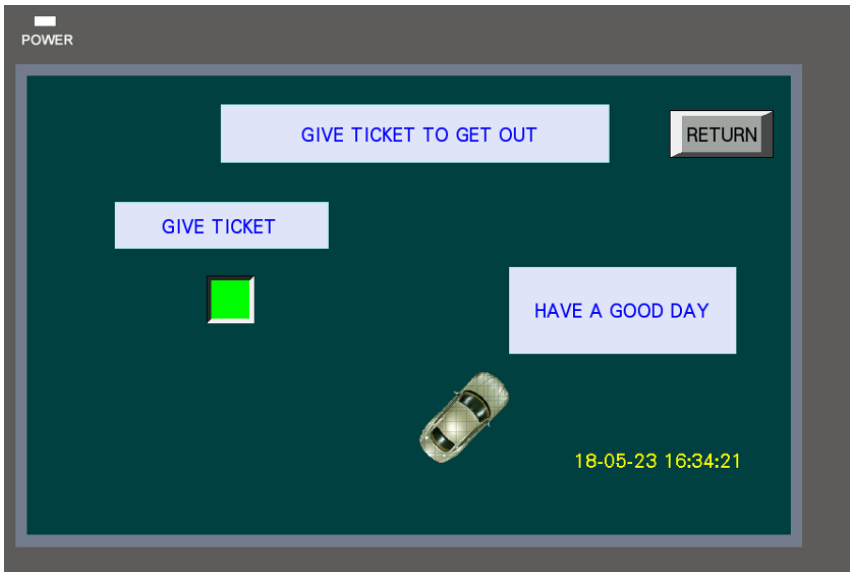

Fig. 3.32. “Give ticket” process

After all, the user will return to “Manual Setting” screen by pressing the “Return” button (Figure 3.33). In this phase, with the aim of simulating the process, the button “Process” should be pressed. There, it can be chosen either the entrance or the exit mode, as it can be shown in Figure 3.34.

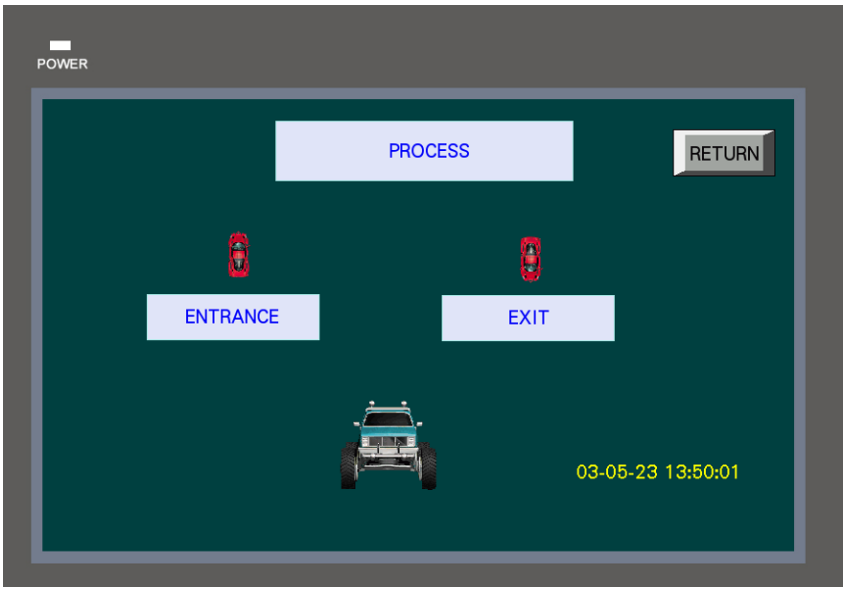

Fig. 3.33. Car process

Fig. 3.34. Entrance process

As can be seen in Figure 3.34, if simulating the entrance process, the user will be able to turn on manually the barrier and the sensors, imitating the entrance action.

Alternately, the exit mode is analogous (Figure 3.35).

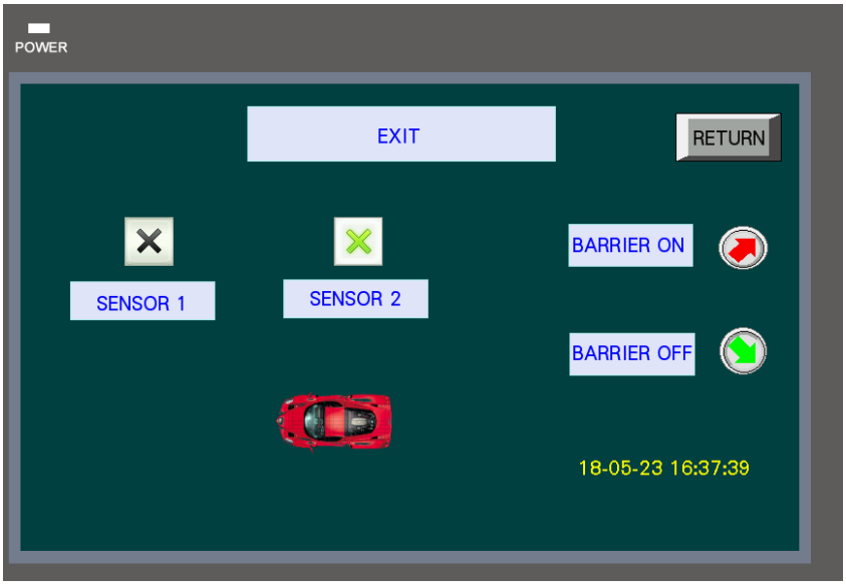

Fig. 3.35. Exit process

Moving to the other section, “PLC setting”, it is referred to show visually the PLC process. In this case, the user will control the external components, which are attached to the PLC, and the entire process will be shown on the HMI screen. In this section, the PLC, HMI and computer will take part, however the HMI will solely have the function of showing the process.

After entering this “PLC setting” section, it will be asked whether the user wants to visualize the entrance or exit process, as it can be shown in the Figure 3.36.

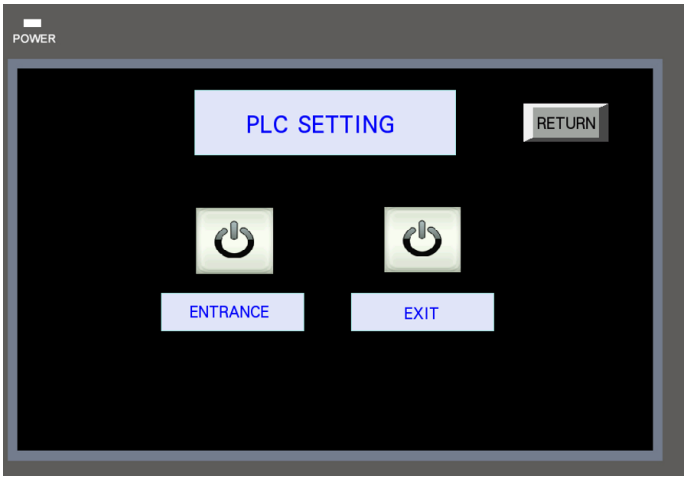

Fig. 3.36. PLC setting screen

First of all, speaking about the entrance action, the HMI screen will show how the car enters the parking, with all the PLC hardware involved. Then, going through this input section, you can see in Figure 3.37 the program without being running. You can easily see all the addresses of each component. On the other hand, some outputs are superimposed, with the aim of simulating different stages. For example, the motor and the barrier appear in different positions. Also, some timers such as the time while the barrier is going up and down, and the timer security, has been implemented. In addition, the car counter.

Fig. 3.37. Entrance process with addressed I/O

After transferring the program to the HMI, the screen will simulate the first stage of the process, as can be seen in the Figure 3.38. At this moment, the car has not taken the ticket yet, and the traffic light is completely green.

In addition, appears two numbers that reflect the required voltage that needs the barrier to go either up or down. The number at the top “Barrier Voltage” shows the range of voltage, between 0-10 V, however, it is displayed in hexadecimal, which is between 0 and A. Moreover, “Barrier Voltage hexadecimal” is related to the hexadecimal number converted from the decimal voltage range.

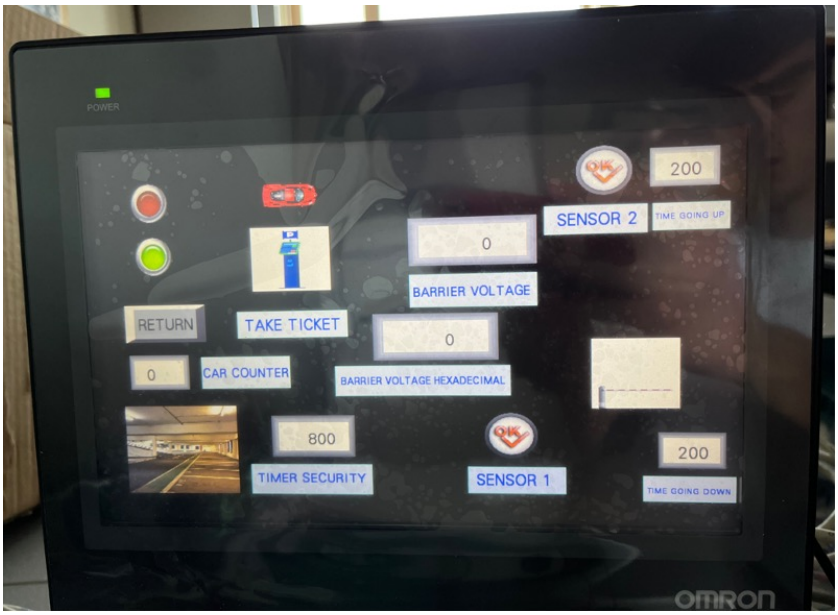

Fig. 3.38. Entrance process first stage

Subsequently, the car will take the ticket by pressing a switch attached to the PLC. Afterward, the HMI screen will show that the traffic light has turned into red. Also, the image of the car will disappear as a result of the car going toward the barrier, Figure 3.39.

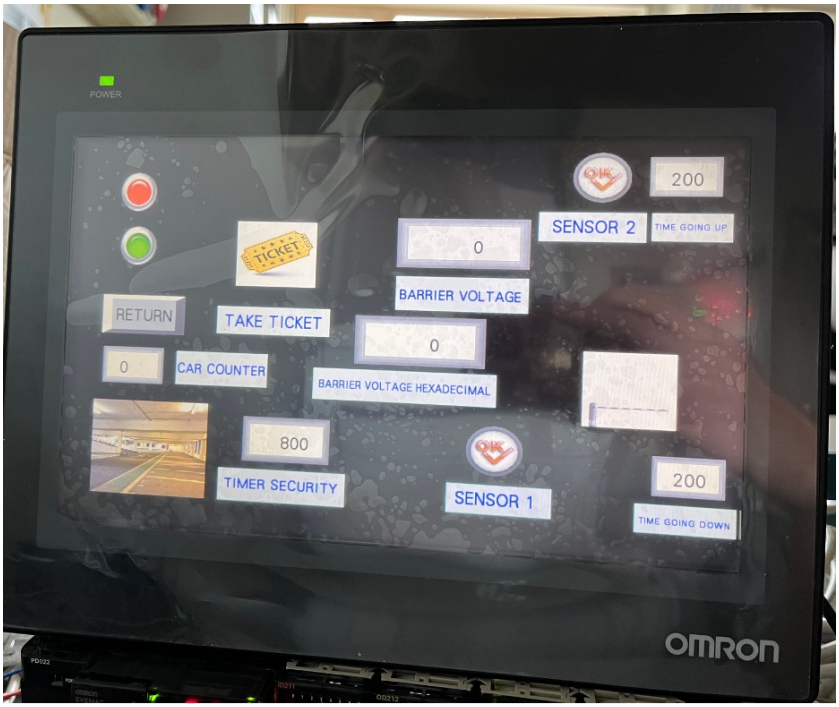

Fig. 3.39. The car takes the ticket in order to start the process

In the next phase, sensor 2 will detect the car approaching the barrier, simulated in the PLC by pressing an attached switch. As we can see in the Figure 3.40, this sensor will energise by turning into green. Moreover, the car's image will emerge on the screen.

Fig. 3.40. Sensor 2 detects the car approaching and the security timer starts running

In the Figure 3.41, it can be shown the rising barrier process. In order to raise the barrier, the user has to regulate the required voltage by adjusting the analog resistor attached to the PLC. This adjustment will be shown on the HMI screen by changing the numbers related to the voltage. Once the analog resistor has been adjusted to a high voltage, the barrier's image will change. Additionally, as a result of rising the barrier, it will emerge a motor image that simulates its rotation and consequently, the car will move forward.

In the Figure 3.42, the barrier's image changed as a consequence of the barrier being completely up. The motor disappeared due to its rotation is over. Also, the car vanished because it is going through the barrier.

Fig. 3.41. High voltage adjustment, the barrier goes up and the motor rotates

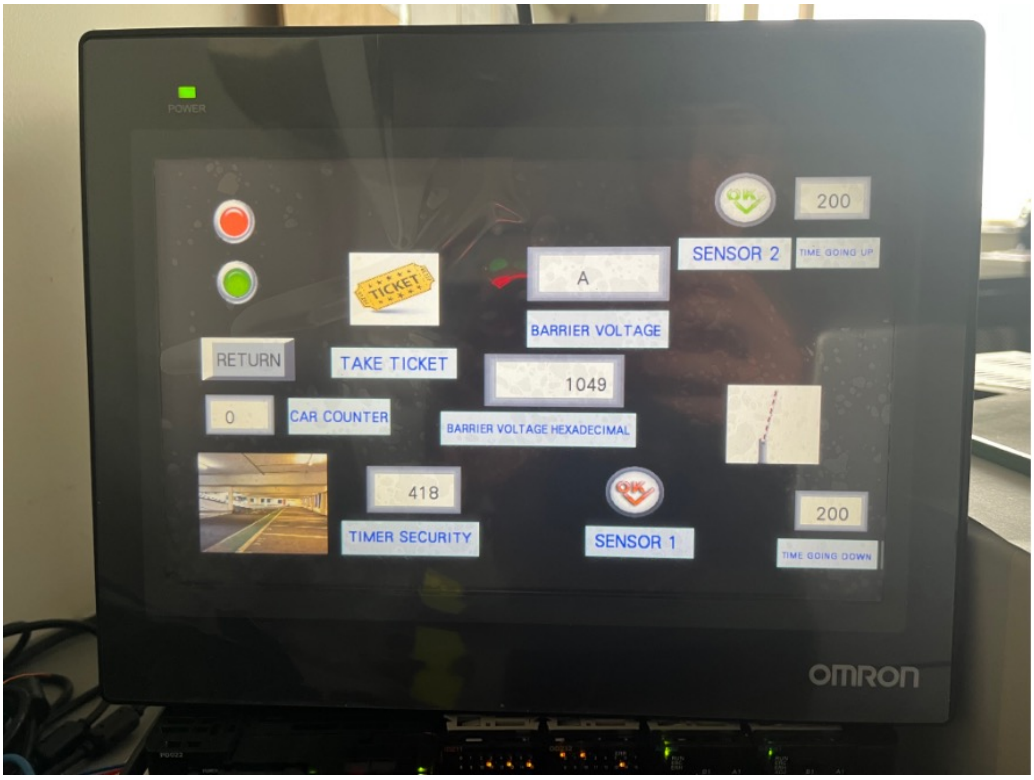

Fig. 3.42. The barrier is completely up and the car goes through it

In the next stage, sensor 1 will detect the car after going through the barrier, simulated in the PLC by pressing an attached switch. As we can see in the Figure 3.43, this sensor will energise by turning into green. Moreover, the car's image will emerge on the screen.

Fig. 3.43. Sensor 1 detects the car after it went through the barrier

In the Figure 3.44, it can be shown the process of making down the barrier. In order to achieve this, the user has to regulate the required voltage by adjusting the analog resistor attached to the PLC. This adjustment will be shown on the HMI screen by changing the numbers related to the voltage. Once the analog resistor has been adjusted to a low voltage, the barrier's image will change. Additionally, as a result of the barrier going down, it will emerge a motor image that simulates its rotation and consequently, the car will move forward. This image of the motor will be different from the one seen before, simulating the counterclockwise rotation.

In addition, it has been set by the analog resistor a low voltage, but not the lowest, with the aim of showing other numbers apart from 0.

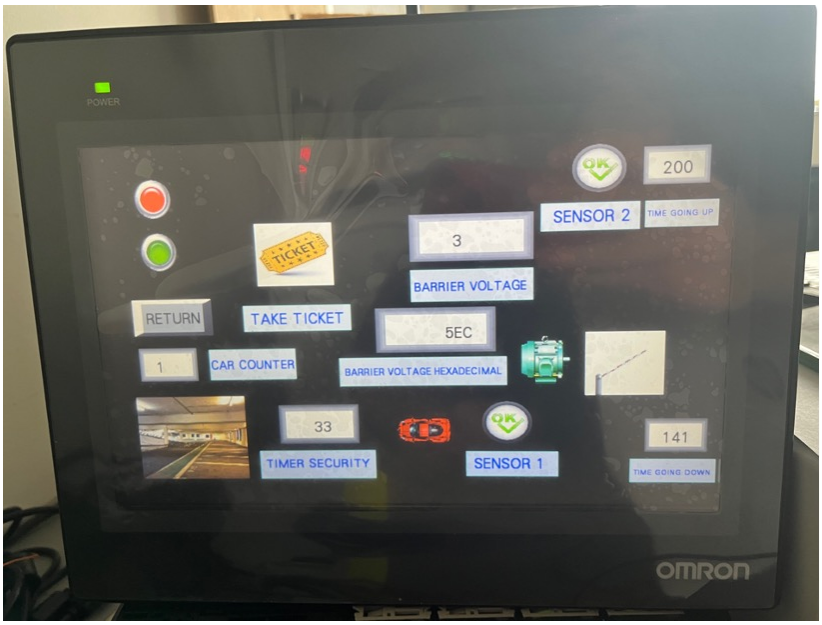

Fig. 3.44. Low voltage adjustment, the barrier goes down and the motor rotates

Then, the process will be reset to the beginning, but the counter will add up to one car. On the contrary, in order to simulate the exit process, it is necessary to press the exit button shown in the Figure 3.45. Finally, the HMI screen will show how the car exits the parking, with all the PLC hardware involved. In fact, this visualization is similar to the entrance process. The addresses of each component are displayed. Also, the first stage of the process can be exhibited in the Figure 3.46.

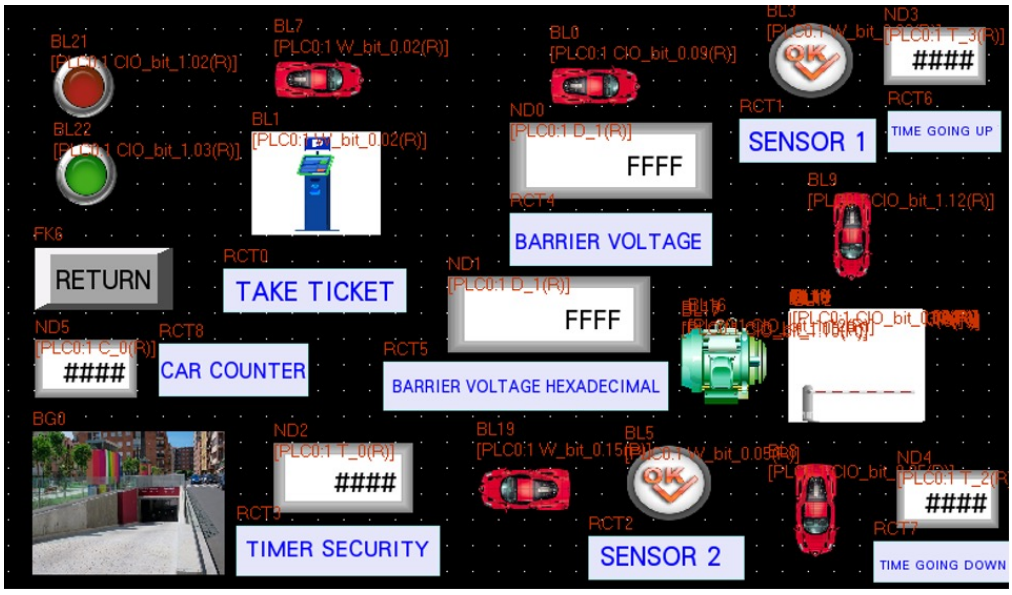

Fig. 3.45. Exit process with addressed I/O

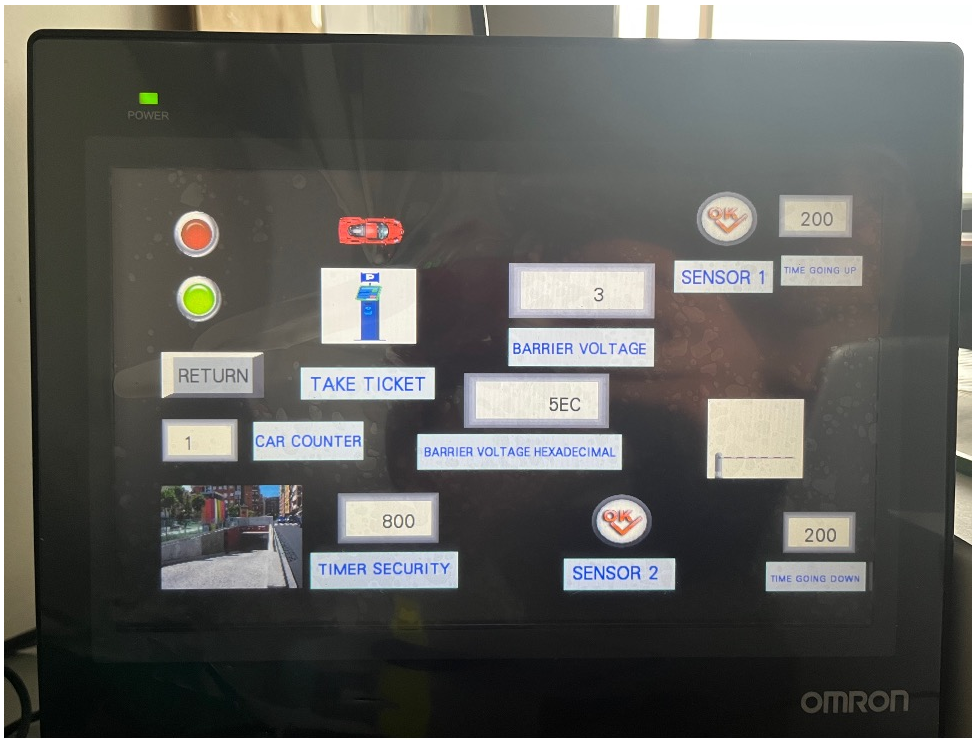

Figure 3.46. Exit process first stage

Basically, in order to achieve this simulation on the HMI screen, the addresses of the PLC software have been linked with the HMI inputs and outputs. For example, some outputs, such as the “Take ticket” image have two different images: the ticket machine when it is OFF and the ticket when is ON. Moreover, related to simulating the car going through each stage, for instance, it has been implemented the car image when it is ON, and no image when it is OFF. Although in the first image car, is on the contrary. The motor images work exactly the same, the motor appears when it is ON, and nothing when it is OFF. Related to the barrier output, many barrier pictures of different phases have been overlapped in order to allow the user to see the correct barrier position at each stage. Also, it works as the motor and car outputs, however, as there are many barrier pictures, new rungs have been programmed into the ladder diagram in order to get more addresses, seen in the Figure 3.47.

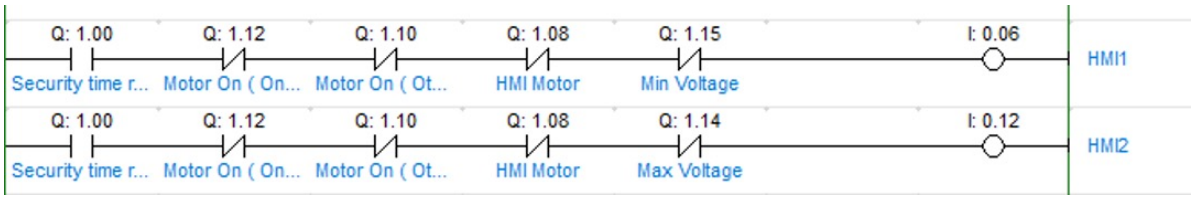

Fig. 3.47. HMI addresses added to the ladder diagram

In Figure 3.48, can be seen the additional rung that has been implemented into the CX-Programmer in order to achieve the last car image.

Fig. 3.48. HMI addresses added to the ladder diagram

The last area, “HMI setting”, is related to controlling the process by the HMI. In this case, the user will be able to simulate the software solely with the HMI. For instance, instead of pressing the attached PLC switches, it will be just necessary to press a button on the HMI screen and will do the same function as the real switch. Basically, the Figure 3.49 can show the functions that the user is able to do by pressing the buttons. It should be emphasized that when the user presses the buttons “Barrier UP” or “Barrier Down”, the motor attached to the PLC will start the rotation clockwise or counterclockwise respectively. In addition, the button “STOP” has been implemented exclusively in this section. It has the target of stopping the process immediately.

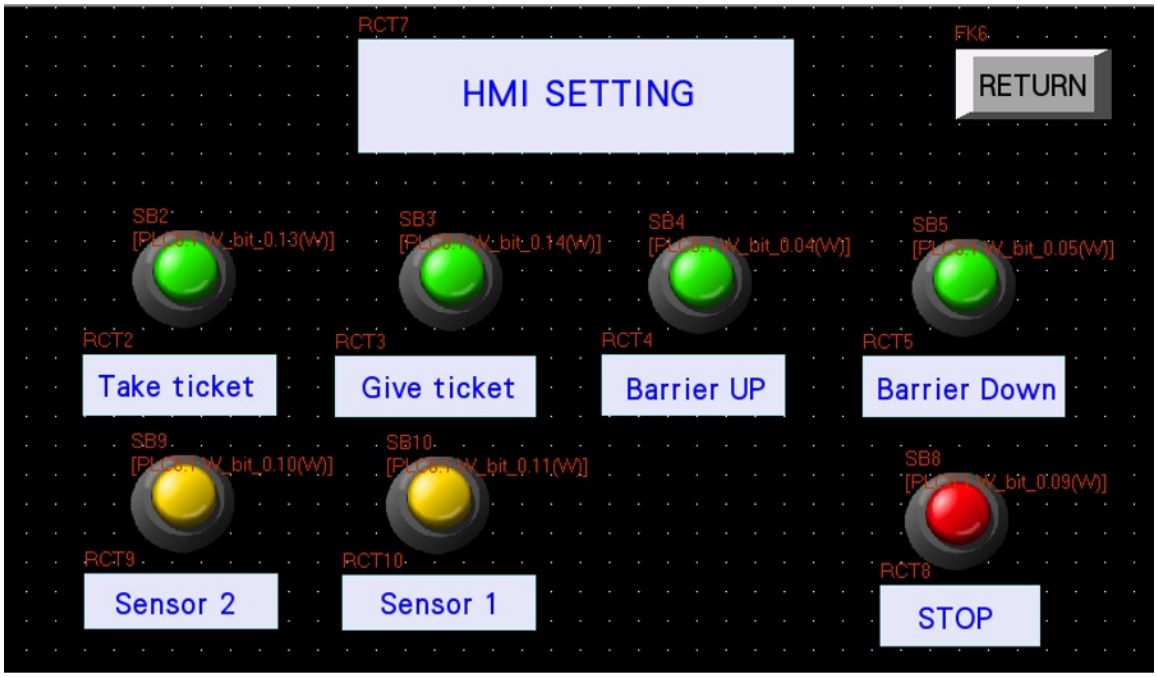

Fig. 3.49. HMI setting screen

For instance, in the Figure 3.50 it can be shown the interaction between the computer, PLC and HMI. When the user presses the button “Take ticket” on the HMI screen, the bit

W0.13 “HMI Take ticket”, on the ladder diagram, will be activated. In addition, the bit W0.01 “Ticket has been taken” will be energised, and the user will be able to move forward to the next step.

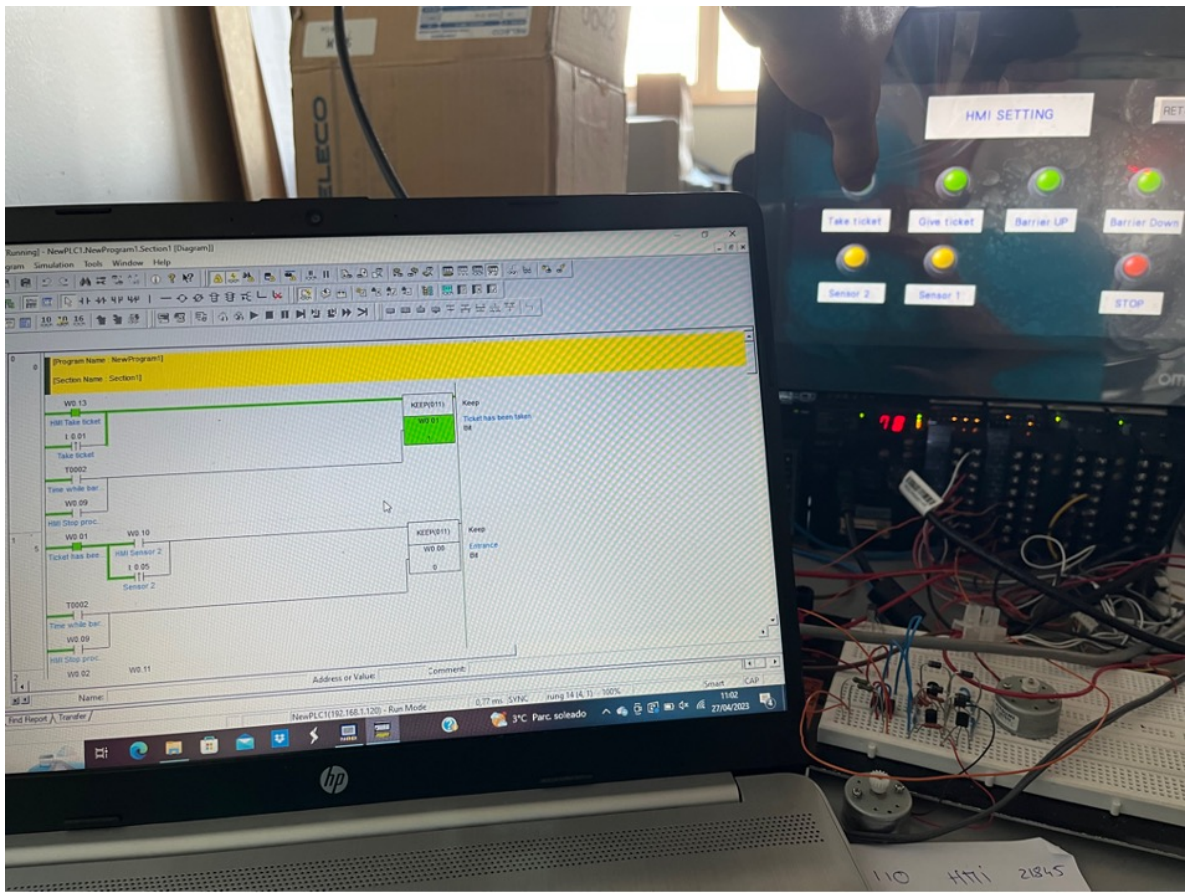

Fig. 3.50. Example of PLC, computer, and HMI interaction by pressing "Take ticket" HMI button

Basically, in order to be able to make the simulation manually with the HMI and interact with the computer and PLC, some additions have been made to the CX-Programmer software. Mainly, adding Ladder Logic OR functions, as displayed in the Figure 3.51. For instance, Ladder Logic OR functions with sensor 2 and HMI sensor 2. An example activating “Sensor 2” from the HMI can be seen Figure 3.52.

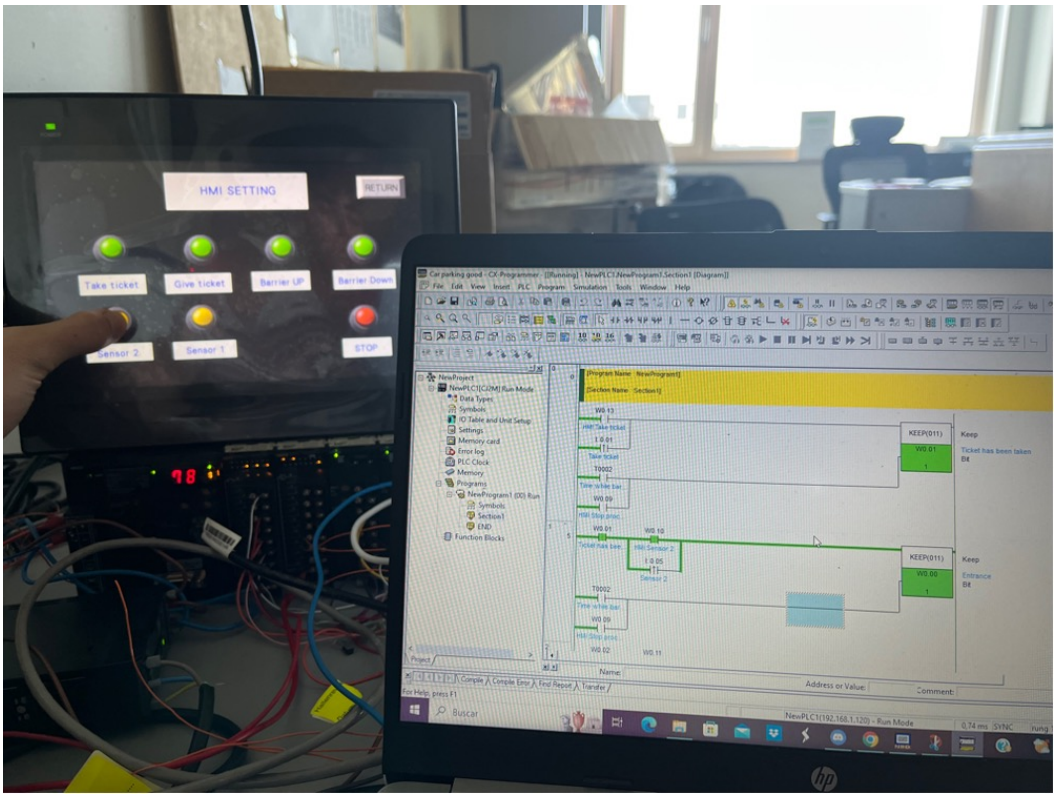

Fig. 3.51. Example of PLC, computer, and HMI interaction by pressing “Sensor 2” HMI button

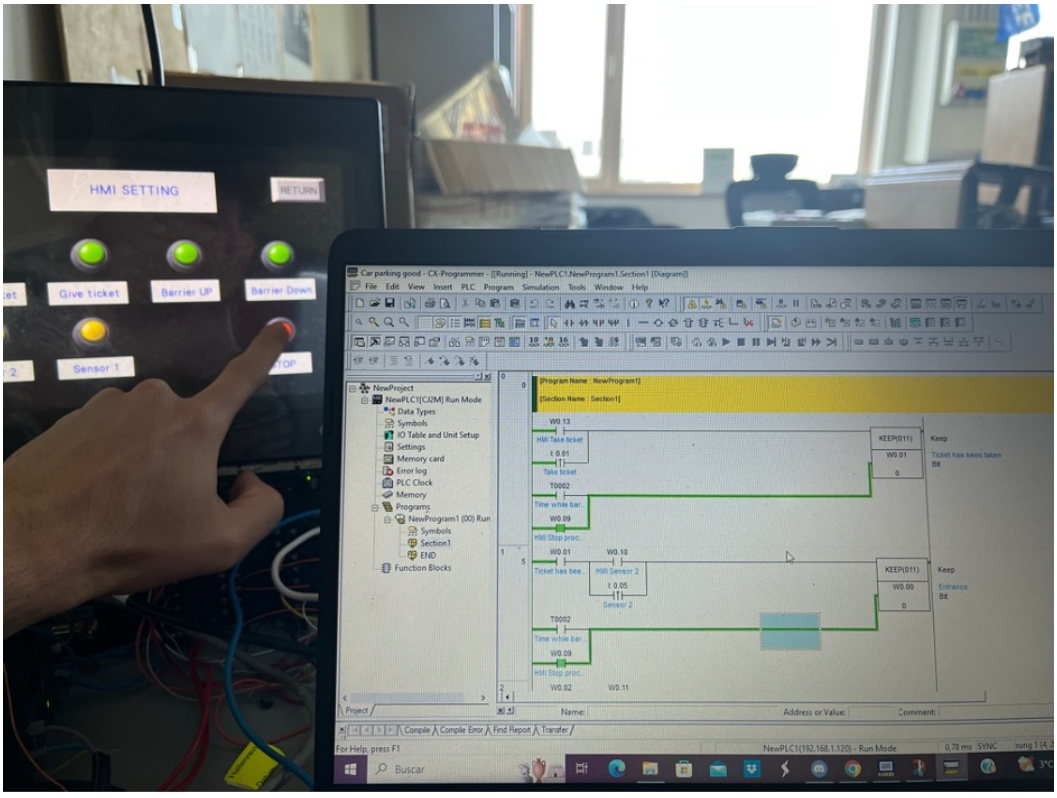

Figure 3.52. Example of PLC, computer, and HMI interaction by pressing the “STOP” HMI button

CONCLUSIONS

This section is devoted to summarising the main findings of this work, as well as suggesting some plans for possible future work.

RESULT FROM PRESENT WORK

In conclusion, the thesis emphasizes the increasing importance of PLCs and HMIs in today´s world, highlighting their role in efficiency, productivity, and enabling effective human-machine interaction. It has been demonstrated the practical implementation of the setup, which provides a safe and controlled environment for testing before deploying the system in actual industrial settings. In addition, this test bench has presented its versatility and scalability by allowing the user to make continuous changes to the software or components, due to its capability to transfer the program to the PLC and HMI quickly. The research conducted in this thesis has provided a comprehensive understanding of the integration between hardware components and software programs. As a result, due to combining theoretical and implementation parts, this project has contributed to a deeper understanding of this topic, which allows future students to acquire the skills needed to contribute to the development and implementation of advanced automation systems.

FUTURE WORK

In order to improve the comprehensive parking garage simulation, it can be added an OMRON camera “ZFV-SC50”. This camera might detect the car´s license plate number and send this information to the PLC, working as a digital input. Moreover, this project could be further refined by adding some analogue I/O examples. For instance, adding either a multimeter that shows the input voltage, which would be positive for the barrier voltage, or a sensor that detects an object approach, which might detect the car and send this information to the PLC as an analog input.

In addition, related to the TCP/IP connection, the computer can be connected via WIFI with the Ethernet network. Apart from that, different real-world simulations can be created, involving different external components.

LIST OF REFERENCES

[1] Petruzella, F.D., *Programmable Logic Controllers*. 4. New York: McGraw-Hill Companies, 2011. 396 lpp. ISBN 978-0-07-351088-0.

[2] Bryan, L.A. and Bryan, E.A., *Programmable Controllers*. 2. Atlanta: Industrial Text Company, 1997. 1035 lpp.

[3] *Rack PLCs guideline. OMRON.* 135 lpp. Available from: https://www.webddigital.com/fabricantes/omron/pdf/plcs/CS1.pdf

[4] Romero-Segovia, V. and Theorin, A., *History of PLC and DCS*. 2012. 21 lpp. Available from: http://archive.control.lth.se/media/Education/DoctorateProgram/2012/HistoryOfControl/Vanessa_Alfred_report.pdf

[5] Langmann, R. and Stiller, M., *The PLC as a Smart Service in Industry 4.0 Production Systems. Dusseldorf:* Faculty of Electrical Engineering & Information Technoloigy, Hochschule Duesseldorf University of Applied Sciences, 2019. 20 lpp. Available from: https://www.mdpi.com/2076-3417/9/18/3815

[6] Benyon, D., *Designing Interactive Systems: A Comprehensive Guide to HCI, UX and Interaction Design.* 3. Harlow: Pearson education limited, 2014. 6504 lpp. ISBN 978-1-292-01384-8

[7] Chapter 7 - Ethernet Connection, *Device/PLC Connection Manuals.* Pro-face. 24 lpp. Available from: https://www.proface.tech/otasuke/files/manual/plc_connection/v70/omr/eomreth.pdf

[8] Pressman, A.I., Billings, K. and Morey, T., *Switching Power Supply Design*. 3. New York: McGraw-Hill Companies, 2007. 792 lpp. ISBN 978-0-07-148272-1.

[9] Chapter 1 - Resistor Fundamentals, *What is a resistor?*. EE Power. Available from: https://eepower.com/resistor-guide/resistor-fundamentals/what-is-a-resistor/

[10] Hughes, A. and Drury, B., *Electric Motors and Drives: Fundamentals, Types, and Applications*. 5. USA: Newnes, 2019. 1368 lpp.

[11] Horowitz, P. and Hill, W., *The Art of Electronics*. 2. Cambridge: Cambridge University Press, 1989. 1125 lpp. ISBN 0-521 -37095-7.

[12] Kennedy, B., *Implementing an Isolated Half-Bridge Gate Driver*. Analog Devices. 2 lpp. Available from: https://www.analog.com

[13] Romanov, V., *How to Read Ladder Logic & Ladder Diagrams*. Available from: https://www.solisplc.com/tutorials/how-to-read-ladder-logic

[14] *CX-Programmer Operation Manual*. OMRON, 2019. 96 lpp. Available from: https://assets.omron.eu/downloads/manual/en/v6/w446_cx-programmer_operation_manual_en.pdf

[15] *Programming Manual*. OMRON, 2007. 1140 lpp. Available from: https://assets.omron.eu/downloads/manual/en/v1/w451_cp1_cpu_unit_programming_manual_en.pdf

[16] *NB-Designer Operation Manual*. OMRON, 2020. 451 lpp. Available from: https://assets.omron.eu/downloads/manual/en/v1/v106_nb-designer_software_operation_manual_en.pdf

[17] *User´s Manual*. OMRON, 2010. 364 lpp. Available from: https://assets.omron.eu/downloads/manual/en/v6/w472_cj2_cpu_units_hardware_users_manual_en.pdf

[18] *Operation Manual*. OMRON, 2009. 487 lpp. Available from:

https://assets.omron.eu/downloads/manual/en/v4/w345_cs1_cj1_analog_i_o_units_operation_manual_en.pdf

[19] Scherz, P. and Monk, S., *Practical Electronics for Inventors*. 2. New York: McGraw-Hill Education, 2016. 1027 lpp.

DATASHEETS

PLC https://www.ia.omron.com/data_pdf/cat/cj2m-cpu3__-cpu1_-md21__ds_e_12_5_csm2201.pdf?id=2712

Motor https://datasheetspdf.com/pdf-file/917203/MABUCHIMOTOR/RF-300EA-1D390/1

Switches differentiated up https://omronfs.omron.com/en_US/ecb/products/pdf/en-ss.pdf

Switch differentiated down https://pdf1.alldatasheet.com/datasheet-pdf/view/333098/OMRON/D2X.html

HMI https://datasheet.octopart.com/NB10W-TW01B-Omron-datasheet-62040884.pdf

Multimeter https://www.libble.eu/metrix-mx-22/online-manual-696641/

Optocoupler https://pdf1.alldatasheet.com/datasheet-pdf/view/43364/SHARP/PC814.html

Power supply https://www.testequipmenthq.com/datasheets/TTI-EX752M-Datasheet.pdf

Router https://downloads.linksys.com/downloads/userguide/1224638367343/WRT54G-v1.1_ug.pdf

Transistor https://pdf1.alldatasheet.com/datasheet-pdf/view/555501/WINNERJOIN/BC557C.html

Diode https://pdf1.alldatasheet.com/datasheet-pdf/view/25755/SURGE/FR102.html

PROGRAMS

CX-PROGRAMMER https://industrial.omron.es/es/products/cx-programmer

NB-DESIGNER https://industrial.omron.eu/en/products/nb

MULTISIM https://www.multisim.com/

LIST OF FIGURES

LIST OF TABLES